# AVIS: AN AUTOMATIC VERIFICATION SYSTEM FOR CAD CIRCUIT-DATA BASE USING PADIKS

By

Ming-Tsair Dai

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1989

TO MY MOTHER

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Dr. Julius T. Tou, the chairman of his supervisory committee, for all the technical guidance, innovative discussion, and continuous encouragement during the course of the work presented in this dissertation. The author would also like to extend his deep appreciation to Dr. John Staudhammer, Dr. Peyton Z. Peebles, Jr., Dr. Leon W. Couch, and Dr. Yuan-Chieh Chow for their friendly help and for their participation on the committee.

Special thanks are given to his colleagues at the Center for Information Research for the stimulating discussions. Thanks are also due to Mr. Rod Wilson for his kind assistance.

Last, but not least, the author thanks his wife for her patience, persistence, and encouragement through the past four difficult years.

TABLE OF CONTENTS

CHAPTER Page

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

AVIS: AN AUTOMATIC VERIFICATION SYSTEM
FOR CAD CIRCUIT-DATA BASE
USING PADIKS

By

Ming-Tsair Dai

December 1989

Chairman: Dr. Julius T. Tou
Major Department: Electrical Engineering

A knowledge-based intelligent system called AVIS for verifying the CAD database generated by an automatic schematic capture system is presented. The domain knowledge following the design blueprint of PADIKS is represented in terms of three primitives: entities, attributes, and relationships. The relationship primitives organize the domain knowledge into the canonical form of content associative tree. The configuration of AVIS' knowledge base reflecting this representation concept is in a sketch-details scheme. A context generator containing a variety of procedures is also included into the knowledge base to facilitate the self-sensing capability of AVIS. This structured knowledge base is able to integrate into

the system the power of symbolic reasoning of expert systems and the computing precision of analytic functions.

Regarding the effective and efficient utilization of the structured knowledge base, an inference control strategy termed SICS is derived from rough set theory in conjunction with knowledge-directed pattern recognition techniques. SICS performs beam search but with the capability of automatic beam-width adjustment. Its inference control consists of two phases: integral analysis and differential diagnosis. The generated solution, conditioned on knowledge structure and the observability of attributes, is complete and concise.

We further augment SICS to include approximate reasoning. We propose an Non-Bayesian method which is based on three foundation concepts: Evidence Space, Belief Characteristic Function, and Uncertain Pattern Matching. The approximate reasoning in SICS, thereafter, is characterized as an uncertainty calculation cycle covering three conceptual levels: attribute, descriptor, and hypothesis. The whole cycle can proceed in a uniform fashion.

The detailed design of AVIS regarding three verification performances (data completeness check, data consistency check, and interconnection verification) is elaborated. Some of the operational results are presented.

# CHAPTER I
# INTRODUCTION

## 1.1 Background to the Problem

The last decade witnessed the stunning growth and success of CAD/CAM (computer-aided design/computer-aided manufacture) technologies. In parallel to the fast development of the micro-electronics enterprise, the territory of CAD/CAM applications has been immensely expanded to a coverage beyond our imagination. Nowadays, almost every aspect of design (especially the VLSI design) and manufacturing processes is based on CAD/CAM. Since IC (integrated circuit) complexity, as predicted by Moore's law (Shapiro and Smith, 1984), doubles approximately every two years, the function of CAD/CAM machines has been dramatically evolving from the cut-and-patch in the first generation to the customer-oriented current generation, and swiftly forward to the next generation integrating artificial intelligence as tomorrow's highly intelligent computer systems as predicted by the researchers in this discipline (Daniel and Gwyn, 1982, Gero, 1987, Horstman, 1983). The wide-spread impact these technologies have created, especially upon the revolution of the electronic industry can never be over-exaggerated.

Despite the marvelous and persistent success, the development of CAD/CAM technologies is by no means complete. We are still able to see some room in the current CAD/CAM facilities for improvement, such as the capturing of input schematics in the front-end design phase (Odawara et al., 1981), the verification of intermediate results during the design process (Wojciki, 1983). In response to the first need, the Center for Information Research at the University of Florida has developed the AUTORED (Automated Electronic Diagram Reading Machine) system (Tou and Cheng, 1983a) with the objective of automating the data entry process which is usually time consuming and prone to error. To cope with the second problem, we present AVIS (Automatic Verification System for CAD database) with the aim of verifying CAD database.

In the past, the capture of electronic circuit diagrams in CAD design process was mainly performed by an interactive editor through a special device such as light pen, tablet (and recently via mouse) (Bayegan and Asa, 1977, Odawara et al., 1981). Contrary to that, AUTORED, making use of image understanding techniques, is designed to read, interpret, and automatically convert the schematic diagrams into CAD database (Tou and Cheng, 1983a). As a consequence, with the elimination of the error-prone schematic editing process, we improve efficiency and productivity of design automation. The AUTORED project was accomplished around 7 years ago. It was a pioneer design not only in demonstrating the integration of a

pictorial system with a CAD machine to reduce manpower, but also in proving the capability of maintaining the vast amount of drawings produced prior to the CAD era. Those drawings representing a priceless treasure of generations' accumulation of knowledge and expertise, have the significant application of creating customized circuit libraries, an important element in CAD systems, for idea browsing and inspiration (Groen and Munster, 1986).

Concentrating on analog design, the prototype AUTORED system organizes the captured schematic into a CAD database containing four files; pin file, line file, junction file, and component file as shown in Figure 1.1 (Tou and Cheng, 1983a, Tou et al., 1987). These four files encode the input schematic into a machine readable format which, under the design objective, can be reformatted to link other simulation packages, such as SPICE (Cheng 1983). However, one problem realized in the prototype AUTORED system is the dependence of accuracy and completeness of the interpretation upon the quality of the raw image of circuit diagrams. Image quality might be significantly deteriorated, due to aged drawings, illumination variation, and several other factors. As an illustration, Figure 1.2 shows a simple example about a differential amplifier. The circuit schematic after image digitization as depicted in Figure 1.3 has its quality deteriorated and involves several ambiguities. Those denotation symbols have feature appearances so similar to

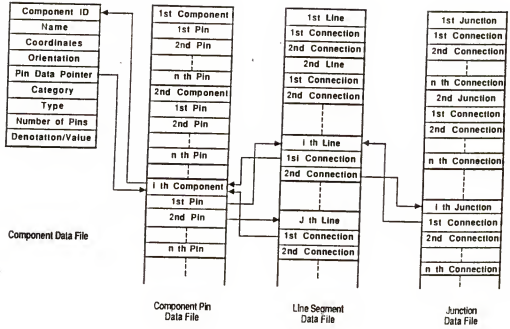



Figure 1.1 AUTORED-generated CAD Database Organization.

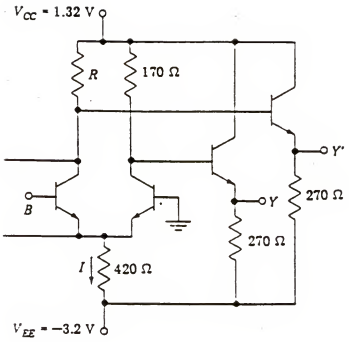



Figure 1.2 Circuit Diagram of Differential Amplifier

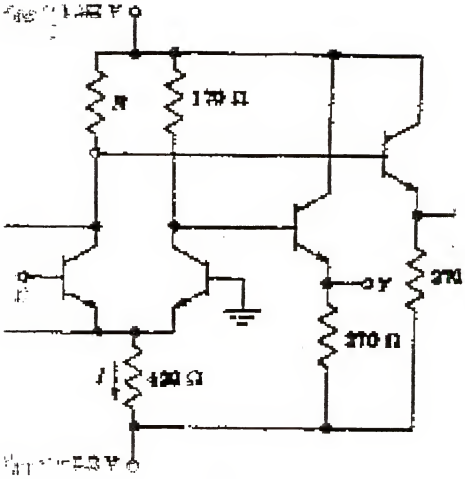



Figure 1.3 Image of Circuit Schematic after Digitization

normal cases. Thus, AUTORED inevitably delivers several misinterpretations. The captured component list displayed in Figure 1.4 contains 20 component symbols (including I/O, voltage source, ground, etc.) but 5 of them are actually superfluous. From experience, the flaws occasionally occurring in the generated CAD data base can be categorized into three major types (Tou et al., 1987):

(1) Incomplete data records- caused by unrecognizable or partially recognized circuit elements which are the results as some parts of the schematic image become smeared or missing after image pre-processing.

(2) Redundant data records- mainly caused by noise not completely removed after image pre-processing, where noise may appear as isolated component symbols, junction points, line segments, or various kinds of superfluous data records.

(3) Erroneous data records- caused by distorted schematics such that the circuit elements are inevitably misinterpreted, and the system ends up with erroneous data records in the generated CAD database.

To guarantee high reliability and accuracy, obviously a verification on the captured database is needed. This concern virtually brings forward the AVIS project, and also delimits our research objectives to include the following three major concerns:

| | X1 | Y1 | X2 | Y2 | - | ORIENT | TYPE |
|---|---|---|---|---|---|---|---|
| | 64 | 4 | 71 | 11 | 0 | 0 | 8275 |
| | 80 | 32 | 91 | 65 | 0 | 2 | 8261 |
| | 45 | 33 | 56 | 65 | 0 | 2 | 8261 |
| | 177 | 56 | 195 | 91 | 0 | 1 | 8258 |
| | 138 | 89 | 156 | 124 | 0 | 1 | 8258 |
| | 210 | 96 | 217 | 103 | 0 | 0 | 8275 |
| | 34 | 113 | 51 | 146 | 0 | 1 | 8258 |
| | 84 | 114 | 102 | 146 | 0 | 3 | 8258 |
| | 188 | 115 | 199 | 146 | 0 | 2 | 8261 |
| * | 200 | 125 | 206 | 134 | 0 | 0 | 8275 |
| * | 207 | 125 | 216 | 133 | 0 | 1 | 8281 |
| | 14 | 126 | 21 | 133 | 0 | 0 | 8275 |
| | 169 | 129 | 175 | 136 | 0 | 0 | 8275 |
| | 149 | 141 | 159 | 173 | 0 | 2 | 8261 |
| * | 171 | 154 | 178 | 163 | 0 | 0 | 8275 |
| | 62 | 155 | 72 | 192 | 0 | 2 | 8261 |
| * | 161 | 156 | 168 | 163 | 0 | 0 | 8275 |
| * | 7 | 210 | 15 | 220 | 0 | 2 | 8261 |
| | 64 | 213 | 71 | 220 | 0 | 0 | 8275 |
| | 108 | 142 | 124 | 157 | 0 | 0 | 8263 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 999 | 999 | 999 | 999 | 999 | 999 | 999 |

FORTRAN STOP

* 5 component symbols are superfluous

Figure 1.4 AUTORED Generated Component File.

(1) Data completeness verification

(2) Data consistency verification

(3) Interconnection verification

## 1.2 Design Concepts of AVIS

Intuitively, the basic functions in data verification include at least diagnosis and correction. The key element is symptom diagnosis which carries out a breakdown of the observations into individual error modes, so that the diagnostic strategy, usually a sequence of decisions, can identify the suspicious symptom modes. We express the whole idea as follows:

$$I * O * F \Longrightarrow D \Longrightarrow (\hat{E}, R(\hat{E})); \quad \hat{E} \in (E_0, E_1, \ldots, E_i, .., E_n)$$

where

I = information data base usually storing system model

O = diagnostic observations

F = functional properties of the diagnosed system

D = diagnostic strategy

$E_i$= $i^{th}$ error mode (i=0,1,...,n)

$\hat{E}$ = diagnosed error mode

R = actions required to correct the diagnosed errors

In this model, a successful diagnosis means the diagnosed error mode $\hat{E}$ can satisfy certain evaluation criteria, such as minimizing the error probability. Quite a few techniques have been proposed with different application concerns. Most of

them, particularly for diagnosing electronic systems, are developed on the basis of a causation model involving well defined casual relationships (Knowles, 1976). The diagnostic strategy is usually the divide and conquer (detection and isolation), since the symptom propagation, following the predetermined casual relationships in UUT (unit under test), can be analytically calculated and monitored. The ATE (automatic test equipment) machines exemplify this diagnostic concept and have gained extraordinary success. However, in case the domain complexity is large or the causation model is no longer available, explicit and algorithmic analyses seems to be difficult, and thus the knowledge-based approaches become attractive (Dejong, 1985, Porter, 1987). Including humans' expertise, heuristic, this new diagnostic concept is versatile and capable of dealing with irregular symptom combination without resorting to the well-defined causation model (or the computationally expensive enumeration). For AVIS, we encounter the similar lack of a strict causation model. In AUTORED, the schematic data are created based on a multiple-pass pattern extraction technique which decomposes a schematic diagram into five sets of drawings such that each pass concentrates and performs only on specific modular function (Cheng, 1983). This approach achieves the advantages generally associated with modular design but somehow scrambles and blurs the global relationships between drawing sets. Therefore the casual relationships in AUTORED's interpretation

mechanism become poorly-structured, case-oriented, and thus untraceable. Due to this concern, for the AVIS project, we also bring in artificial intelligence techniques, and designate AVIS as a knowledge-based expert system which, as a matter of fact, also represents our continuous dedication to the development of intelligent CAD/CAM systems (Tou and Cheng, 1983a, Tou et al., 1984, Tou and Huang, 1984, Tou et al., 1987).

Expert systems have been introduced to us for a few decades. Research in this field has had many important successes (Kowalik, 1986b, Waterman, 1986). The potential power of systems which can replicate and autonomously apply expensive human expertise has led to a worldwide effort to explore and extend this technology (Hayes-Roth et al., 1983, Charniak and McDermott, 1985, Waterman, 1986). Various design methodologies have been proposed. Among the well-known, there are the rule-based approach, procedural system, frame-based approach, etc. (Barr and Feigenbaum, 1981) However, in AVIS' target domain, we find at least two subtleties which make AVIS' design specifications distinctive from conventional expert systems:

### 1.2.1 Uncooperative Working Environment

Generally, in diagnosis or verification, the system infers malfunctions from observation and deduces underlying causes from behavioral irregularities, where the diagnostic

observation are supposed to be honest and reliable, a key factor to successful operation. Alternatively speaking, in a working diagnostic system, the data base should hold pertinent and trustable information to support the inference mechanism of the system for certain pursuits in the knowledge base. Under the tight coordination of the inference mechanism, the data base and the knowledge base cooperatively deduce new facts and gradually solve the problem. However, in AVIS, this close cooperation is not quite obvious. Instead of fully trusting the observations (data records) made in the CAD data base, AVIS is asked to use its knowledge to justify, even revise the observations (data records). This type of relationship we term uncooperative. Even worse is the existence in AVIS' target domain of a nonmonotonic phenomenon which seems to be inevitable because the symptom features of schematic images are composed from the data of either local type, or global type, or their combination. The intertwined relationships sometimes create unexpected interference; therefore, it is not impossible for the system to withdraw formal conclusion after reaching a new verification decision. Following these concerns, AVIS' functional capabilities should be different and beyond the conventional expert systems, even the lately discussed expert data base systems where only the coordination of data transfers among a large group of expert systems is the main interest (Kerschberg, 1986).

1.2.2 Coupling Symbolic Reasoning and Numerical Computing

Combining together symbolic reasoning with formal computing into expert systems has received growing attention (Kowalik, 1986a). It is viewed by researchers as a logical next step in upgrading the performances of expert systems. AVIS demonstrates this trend. The verification knowledge AVIS needs is mainly excerpted from image processing, circuit theory, device behavior, design conventions, etc., all shown in highly abstracted symbolic expressions. However, the data stored in AUTORED-captured data base are encoded circuit schematics including line interconnection, component location, etc., which are conceptually too low level to lend themselves to direct utilization. In other words, to function between these two incompatible levels, the system should be able to extract significant features or attributes of the data from a background of irrelevant details. This necessity enforces AVIS to equip certain self-sensing capabilities for conducting quantitative computation, which is also beyond the conventional scenario of expert systems.

## 1.3 Previous Work

The verification of AUTORED-generated data base was tried before (Tou et al., 1987) and Figure 1.5 shows the functional organization of that verification system built at this research center. It mainly contains four components: error identification, knowledge base, inference mechanism, and

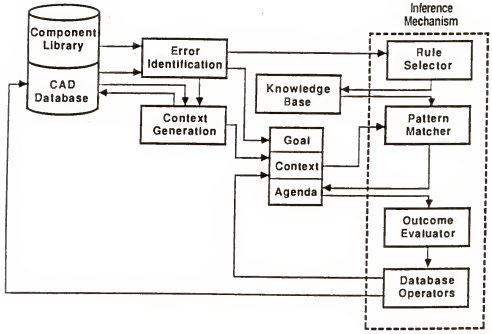



Figure 1.5 Function Flow of A Knowledge-based Verification System

working memory. The verification knowledge extracted from design conventions and circuit theories are stored in the knowledge base, and they are represented basically by the formate of production rule, for instance the following ones for identifying transistor terminals:

```
Rule 2.1.1.1.1.1
  IF      (the component is a bipolar transistor)
          (the arrowhead is located on terminal 3)

  THEN    (terminal 2 is collector)
          (terminal 3 is emitter)

Rule 2.1.1.1.1.2
  IF      (the component is a bipolar transistor)
          (terminal 3 is connected to ground)

  THEN    (terminal 2 is collector)
          (terminal 3 is emitter)
```

The whole set of knowledge, however, has been carefully partitioned into suitable sub-modules based on error types, and they are acquired by a backward reasoning policy.

Four elements (rule selector, pattern matching, outcome evaluator, and database operators) actually constitute the inference mechanism. When the type of error in the captured CAD database is identified, the influence rule selector fetches a subset of rules from the knowledge base which are associated with the error type. The identified error information is also used by the context generator to select a subset of context generation rules for fetching erroneous

data records as generated context to be temporarily stored in the working memory. The error type information, furthermore, establishes a goal for the inference mechanism whose inference principle is iterative matching of generated contexts with appropriate rule patterns in the knowledge base. If more than one rule is matched, the outcome evaluator will choose the one with highest score of similarity measure and the selected rule, thereafter, will activate a number of database operations specified in the action part of the rule. The activated database operations will change the contents of the working memory and/or the CAD database, and thus, complete a verification cycle. This system demonstrates its strength particularly in component related verification operation.

However, a few problems are encountered in this system, mainly caused by the backward reasoning strategy and the rule firing process. Even though the domain knowledge has been properly partitioned and organized, the backward control strategy somehow makes the solution search opportunistic. This is explicitly exposed in the functional coordination between rule selector, pattern matcher and outcome evaluator (i.e., fetching a subset of rules and firing the one based on the matching result between diagnostic contexts and rule patterns). This inference process, strictly speaking, has the image of traditional rule-based approach but with the improvement of an organized rule base. Rule-base under backward chaining generally operates on a data-driven scheme.

Obviously this system has difficulty in dealing with the uncooperative working environment mentioned in the last section, simply because the faulty data might cause the rule selector to pick up the wrong rules, and confuse the pattern matcher and outcome evaluator to fire unnecessary rules. Furthermore, the nonmonotonicity in the problem domain might also hinder the inference mechanism due to the lack of truth maintenance capability in this system. Moreover, the knowledge base of expert systems, as it is all known, should be domain-specific. The verification knowledge of this system is drawn largely from design conventions, and circuit theories might not be enough to deal with AUTORED-oriented image problems, such as superfluous lines, junction type errors, etc. On the other hand, the uncertainty issue regarding the impression in knowledge and data is not touched in this system.

Due to these considerations, instead of tailoring this system or other well-publicized methods, such as frame-based expert systems (Waterman, 1986, Shortliffe, 1976), to suit our requirements, we turn to the design concept of PADIKS, the acronym for Pattern-Directed Knowledge System, to accomplish this project. PADIKS, an alternative for expert system design, was conceived by Tou some fifteen years ago (Tou and Sutton, 1974, Tou and Depree, 1978). It bases its design philosophy upon a structured knowledge base coupled with the knowledge-directed pattern recognition technique for knowledge seeking and problem solving (Tou, 1985, Dai and Tou, 1988). This

design approach, thanks to the clustering concept embedded in its inference logic, has been proved particularly effective in categorical type reasoning (i.e., dealing with the problem domains with classificatory properties). Two intelligent systems: MEDIKS; a medical knowledge system for diagnostic consultation and clinical decision making (Tou and Depree, 1979, Tou, 1978a, Tou, 1978b, Chang and Tou, 1984), and APRIKS; a knowledge-based expert system for application in agriculture (Tou and Cheng, 1983b) exemplify this benefit. In AVIS' problem domain, we can also find significant taxomonic properties. For instance, a data error mode can be classified into three categories; line, junction, and component. Following the symptom appearances, it can be further classified into denotation line, component line, connection line, etc., and further into type-1 denotation line, type-2 denotation line, and so on. Those classificatory features are the viable clues for realizing PADIKS' design concept. For the AVIS project, we drew on the experience from these two example systems; MEDIKS, and APRIKS, and further expand and consolidate the PADIKS design concept into a full-fledged framework adequate for AVIS application.

Figure 1.6 shows the system architecture we propose for AVIS. It contains three major parts; a structured knowledge base, an inference mechanism and a validation controller. The structured knowledge base, following PADIKS approach, is configured into three portions: knowledge sketch, knowledge

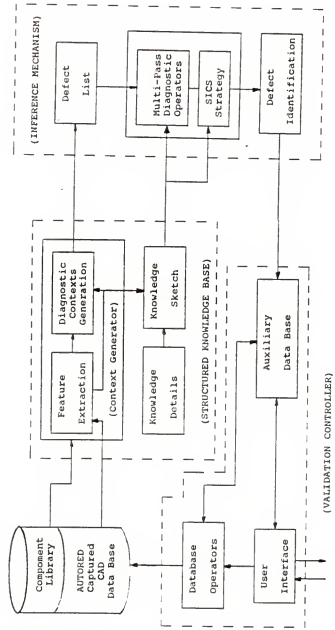



Figure 1.6 System Organization of AVIS

details, and context generator. The knowledge sketch represents the outlines of the domain knowledge for defect localization and identification. Knowledge details, on the other hand, contain various data files in order to support the function of the knowledge sketch. The significant improvement is the organization of the domain knowledge in the knowledge sketch into the structure of content associative tree, which, in conjunction with pattern recognition techniques, facilitate a systematical diagnosis (in contrast to the opportunistic backward chaining process in the early design). The context generator in association with a number of analytic routines is responsible for feature extraction and diagnostic contexts generation. It facilitates the self-sensing capability of the system, which, as mentioned in the last section, is necessary for AVIS to perform as an automatic system. Even though the knowledge base has been partitioned into three different units, the coordination between these three units, in fact, is natural and efficient, because all the control parameters are pattern-based (abstracted knowledge patterns). This is another good aspect in this design. Moreover, the domain knowledge in AVIS is acquired from a broader basis including design conventions, circuit theories, and also from the AUTORED experience following an empirical knowledge acquisition process. Therefore, the knowledge contents are more specific, AUTORED-oriented, and effective.

The inference mechanism is the one actually controlling the process of knowledge seeking and defect diagnosis. Its reasoning process is based on the SICS inference control strategy, which is an adaptive beam search algorithm we developed for the structured knowledge base. A Non-Bayesian framework for approximate reasoning is also devised to deal with the uncertainty issue in both knowledge and data. Moreover, the multi-pass diagnostic policy incorporated in this inference mechanism further takes care of the nonmonotonicity properties of the problem domain.

The validation controller is designed to make compensation on the diagnosed flaws. It contains three elements, auxiliary data base, user interface, and database operators. The auxiliary data base holding the intermediate verification results also helps the inference mechanism to tackle the nonmonotonicity issue. Inside the user interface, a partnership concept is included to resolve the problem created by the uncooperative working environment formed in the problem domain. The database operators, on the other hand, carry out the validation instructions and modify the data base contents.

The detailed design principles and techniques regarding the functions of those system elements mentioned above are the main objectives in this dissertation.

## 1.4 Organization of Dissertation

In Chapter II, we examine the design of AVIS' structured knowledge base. We first give an overview about "knowledge," the key ingredient of expert systems, regarding its representation and utilization. Performances and constraints for various representation skills are discussed comparatively. From that, we contrast out the distinctiveness and superiority of the structured knowledge representation adopted in AVIS, including the representation of knowledge in terms of entities, attributes, relationships, three kinds of primitives, the organization of knowledge into the structure of content associative tree, the configuration of AVIS knowledge base in terms of knowledge sketch and knowledge details, context generator, and so on. The actual generation of AVIS' knowledge base including an empirical approach for acquiring AUTORED specific knowledge is also interpreted.

Chapter III discusses the issue of knowledge utilization, namely inference control. After reviewing some commonly used inference techniques, we elaborate an improved control metaphor called SICS, short for squeeze inference control strategy, which is developed for accurate diagnosis on AVIS' structured knowledge base. It is a beam search with the capability of adaptively adjusting its beam-width via a data-driven fashion (i.e., following the variation of system state). Its inference principle is first to localize the best upper bound of the plausible defect modes from the power set

of hypotheses preenumerated in the problem domain, then, it tightens the boundary via creating the best lower bound conditioned on the knowledge structure and diagnostic observation. The inference process will be terminated when these two bounds become equal. Conditioned on the knowledge base and the observability of symptoms, SICS' clustering capability can be proved, based on the rough sets theory (Pawlak, 1982), to be complete and concise, thus, backtracking, in theory, is not necessary. The implementation of SICS in terms of three distinctive processes--integral analysis, differential diagnosis, and termination check--is investigated. Another advantage of SICS regarding multi-faults diagnosis is also explored. In dealing with multi-faults diagnosis, even though the computation resources versus a large number of active hypotheses tends to be a difficult issue along with most inference strategies with sequential control character, the upper bound concept of SICS is able to overcome this difficulty. The sought solution, in fact, also satisfies the principle of parsimony which demands the solution be complete and concise without irrelevant guesses or redundant suggestions.

In Chapter IV we augment the capabilities of SICS via incorporating uncertainty calculus into its inference strategies. We first discuss the controversy, Bayesian or Non-Bayesian, in approximate reasoning. Then we describe a Non-Bayesian uncertainty method specializing on the inference

logic of SICS. The foundation of this approximate reasoning approach involves three concepts: evidence space, Belief Characteristic Function, and uncertain pattern matching. Evidence Space facilitates an intuitive but effective means for reporting contradictory information. Belief Characteristic Function (BCF) is an operator which lessens the ambiguity reported in the evidence space. The uncertain pattern matching, on the other hand, is the guidance for evidence combination. On this foundation, the mechanism of the approximate reasoning can be formulated into an uncertainty calculation cycle involving three conceptual levels: attribute, descriptor, and hypothesis. The whole calculation cycle can be carried out in a coherent manner in the sense that the concepts of the evidence space, BCF function, and uncertain pattern matching can be alternatively employed at the three conceptual levels in a uniform fashion.

After the exclusive and comprehensive discussion of the key components of AVIS regarding the structured knowledge base design, the SICS inference control, and approximate reasoning, in Chapter V, we discuss the integration of the overall system including the validation controller, and some customized techniques, such as multiple-pass diagnosis, human partnership, which are included to resolve the problems of the nonmonotonicity and the user-uncooperative factor in AVIS' problem domain. An example demonstrating AVIS capability for data completeness and consistency check is also presented. A

different approach for dealing with the third verification objective; interconnection, is also examined in this chapter. Two operators, node verifier and path verifier, in conjunction with the technique of network excitation labeling are devised to accomplish this task. A practical example in this category is worked out for performance demonstrations.

Chapter VI summarizes the major accomplishments reported in this dissertation and provides suggestions for further research.

In APPENDIX we examine some issues regarding knowledge organization, and also a model for the generation of knowledge sketch.

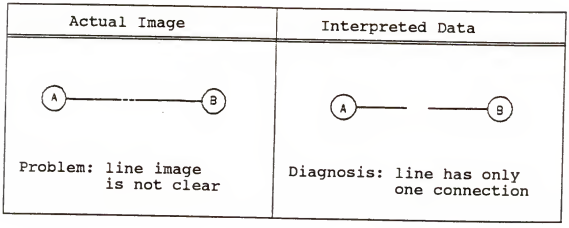
# CHAPTER II
# AVIS STRUCTURED KNOWLEDGE BASE DESIGN

## 2.1 Errors In Captured Schematic Data

Circuit diagram recognition and interpretation of AUTORED makes use of basic geometric features such as shape, length, terminals, and number of pixels to identify circuit elements. The interpretation result is dependent on the quality of the raw image of the circuit diagram, and any noise or distortion of the image is likely to cause interpretation errors as we have briefly pointed out in Chapter I. The errors in the captured schematic data can be categorized into three major types: incomplete data records, redundant data records, and erroneous data records, and thus AVIS' verification performances include completeness check, consistency check, and interconnection verification. Type 1 error may be detected by completeness check. Any missing fields of the data records signify certain errors. For example, the automated recognition of connection lines in a circuit diagram may result in the following kinds of defects: broken lines, missing lines, denotation lines, junction lines, component boundary lines, and so on. Figure 2.1 illustrates two cases of recognition errors in this class. In case 1, a broken line is recognized as two lines. Each line now has only one connection. In case

TYPE 1 ERROR (INCOMPLETE DATA RECORDS)

Case 1: Broken Line




Case 2: Missing Line

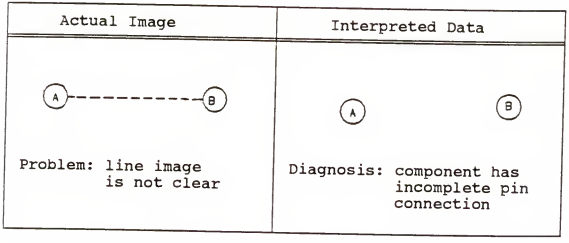



Figure 2.1 Detecting Errors By Completeness Verification

2, a missing line causes the components to have missing terminal connections. These errors, therefore, can be detected by examining the data entries of the related circuit elements.

For the type 2 error, redundant data records, they may be detected by finding certain inconsistencies (i.e., consistency check). Figure 2.2 shows two examples about superfluous junctions, one with connection to a dangling line and the other with clustered junction connections. These errors can be detected based on the violation of certain conventions in AUTORED system, such as

"A LINE SHOULD HAVE TWO CONNECTIONS."

"JUNCTION IS CREATED AT THE INTERSECTION OF TWO LONG LINES."

Data consistency verification is designated to this purpose.

Type 3 error, erroneous data records, may be detected by examining the interconnection relations. Figure 2.3 depicts two recognition errors involving a mis-recognized component and a superfluous component caused by the poor quality of image, and they are detected based on certain electrical properties as shown in Figure 2.3. The interconnection verification is included in AVIS for this purpose.

As is evident from the examples just mentioned, to verify those errors domain knowledge is provided for interpretation and inference. In AVIS, domain knowledge is defined as the knowledge required to verify the errors in the interpreted

TYPE 2 ERROR (REDUNDANT DATA RECORDS)

Case 1: Superfluous Junction

| Actual Image | Interpreted Data |
|---|---|
| denotation  | dangling line  |
| Problem: mis-interpreting denotation symbol | Diagnosis: junction should connect to other junction |

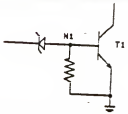
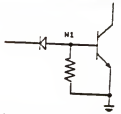
Case 2: Clustered Junction

| Actual Image | Interpreted Data |
|---|---|
| denotation  | clustered junction  |
| Problem: mis-interpreting denotation symbol | Diagnosis: junction arms should be long line segments |

Figure 2.2 Detecting Errors By Consistency Verification

TYPE 3 ERROR (ERRONEOUS DATA RECORDS)

Case 1: Component Mis-Recognition

| Actual Image | Interpreted Data |
|---|---|
|   Problem: Zener diodes are usually recognized as regular diodes |   Diagnosis: no base current can flow into transistor |

Case 2: Superfluous Component

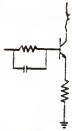
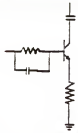
| Actual Image | Interpreted Data |
|---|---|
|  Problem: noise might be mis-recognized as component |  Diagnosis: DC current has been cut off from transistor |

Figure 2.3 Detecting Errors By Interconnection Verification

circuits based on both circuit-level reasoning process and AUTORED-oriented image properties. The circuit-level reasoning process depends on either circuit theories or design expertise to determine if a certain situation is going to happen. This kind of knowledge concerning circuit theories or design conventions is termed general knowledge. They can be acquired from several sources: text books, design manuals, the human's experience and expertise through consultation with circuit designers. Consider the following statements:

> "INPUT TERMINALS IN CIRCUIT DIAGRAMS ARE USUALLY PLACED AT THE LEFT-HAND SIDE WHILE VOLTAGE SOURCE IS LOCATED AT THE TOP OR BOTTOM OF THE CIRCUITS."

> "JFET MIGHT MIX WITH BJT IN CIRCUIT DESIGN, WHILE MOS DOES NOT."

These two are related to design habits and may be useful in identifying circuit elements.

> "IN LINEAR DESIGN, THE COLLECTOR CURRENT OF BJT TRANSISTOR IS A FUNCTION OF ITS BASE CURRENT."

> "DC BIASING CURRENT IS NECESSARY FOR TRANSISTOR OR ACTIVE DEVICES TO FUNCTION PROPERLY."

This kind of knowledge regarding electrical properties is important in interconnection verification.

Contrary to the general knowledge, the AUTORED-oriented image properties provide the viable low-level feature information concerning the irregularity of circuit element appearances, such as broken-lines and denotation-junctions. This kind of knowledge is termed specific knowledge and it can be acquired only from the AUTORED experience. For instance, the following two statements:

> "A DENOTATION LINE USUALLY APPEARS AS AN INTERNAL LINE WITH ITS LOCATION CLOSE TO A COMPONENT BLOCK."

> "A COMPONENT BOUNDARY LINE MUST RESIDE INSIDE A COMPONENT BLOCK WITHOUT ANY NORMAL JUNCTION CONNECTION."

This type of knowledge is specific and it comes from the functions and conventions of AUTORED. For instance the last case is actually obtained from the blocking algorithm of AUTORED (Cheng, 1983). Specific knowledge is the most important knowledge for verifying all kinds of superfluous symbols. It is vital to the data consistency and completeness check. In section 2.4.1, we shall discuss an empirical method for acquiring this kind of domain knowledge.

The accumulation of domain knowledge mentioned above brings into prominence a practical question that is how to adequately represent this diversity of knowledge. Alternatively speaking, in order to exhibit intelligent

behavior, schemes have to be developed for incorporating domain knowledge into the computer programs of AVIS. Since what needs to be represented is known a priori, the task becomes one of figuring out how to encode the knowledge and information in the system data structure and procedures. This concern, in fact, drives us into the heart of the vigorously debated subject of knowledge representation.

## 2.2 Knowledge Representation Formalisms

Knowledge representation is the most active area of AI research (Brachman and Levesque, 1985). The techniques and theories have undergone rapid change and development in the last few years, but the understanding of different representation schemes that researchers have devised seems to be still incomplete. "How to represent knowledge, especially in a coherent and effective manners?" It is a question still facing the knowledge system designers. Technically speaking, for this specific concern we want to know (Gevarter, 1985):

(1) How is the knowledge to be represented, so that it is both usable by the system and comprehensible to human beings? This question regards the rarefaction of a real world domain into abstracted and stylized version acceptable to both machines and humans.

(2) What architecture should the knowledge base be built upon so that the knowledge can be ready to process, transfer, and grow. This question regards the creation of proper

data structure to host the internal representation of knowledge.

The former issue had been widely discussed in the past few decades without leading to a unique solution. Without difficulty we can accumulate a list of ideas, such as (1) logic (Nillson, 1980), first developed by the ancient philosophers as a formal treatment of knowledge about thought, but recently reexplored and extended into programmable logic framework (Robinson, 1965) which was viewed by some AI pioneers as the only formal tool for analyzing human knowledge, (2) semantic network (Findler, 1979), a once popular means for representing declarative and relational descriptions, (3) production rule (Watterman and Hayes-Roth, 1978), generally acknowledged as the most intuitive way to encode human knowledge, and has gained many supporters in rule-based systems, (4) frame (Minsky, 1975), originally proposed to describe stereo-typed situations but recently gained momentum and popularity in representing knowledge, (5) analogical representation (Charniak and McDermott, 1985, Nillson, 1980), often used in the image processing and computer vision research, (6) script (Schank and Abelson, 1977), a modified frame-like structure which finds applications in dynamic planning, (7) semantic primitives (Schank, 1975), developed for the purpose of natural language understanding, and etc. This variety of techniques, in general, can be categorized into declarative versus procedural

along one dimension, or syntactic versus semantic along the other (Rich, 1983). For the declarative methods, such as semantic network, knowledge is represented as a static collection of facts accompanied by a small set of general procedures for manipulation. In the procedural methods, the bulk of knowledge is represented as procedures without explicit indications of using it. Planner (Hewitt, 1971) is a good example. Along the other dimension, in syntactic representational schemes, not much concern is given to the meaning of the knowledge that is being represented, instead, syntactic formula is the main subject of interest. Hence, whatever the information contents might be, they are all manipulated by simple, uniform rules. Logic precisely exemplifies this standpoint. To the other extreme, in semantic systems there is no uniform form exists. Every aspect of representation corresponds to a different piece of information, and thus the inference rules are correspondingly complicated. Semantic network, frame, etc., all stress this viewpoint. Most representation techniques, however, seem to fall into the spectrum of syntactic-semantic, or declarative-procedural, with certain degree of combination from the two extremes. Examples include KRYPTON (Brachman, Fikes and Levesque, 1983), KL-ONE (Brachman and Schmolze, 1985). Based on different representation theories, the architecture of the knowledge base are corresponding different.

## 2.3 Structured Knowledge Base: PADIKS Approach

Domain knowledge in AVIS is not represented as scattered rules, or clauses, instead, based on PADIKS concept (Tou, 1985), they are explicitly organized into a structured format. The representation principle can be characterized by this slogan equation:

Representation = Contents + Access

where the "+" is the ingredient denoting the effectiveness and availability of knowledge. It implies certain strategies which transform the real world problems into stylized notations of task states and provide directives to acquire a successful solution, or to obtain guidance in constructing it. This representation scheme tends to purse knowledge representation with explicit utilization. Within this representation framework how to encode individual knowledge chunk is not the only concern. How to represent the whole knowledge domain into a working unit becomes the key objective. In the following, we first interpret the representation principles including the knowledge rarefaction via a pattern-based approach, the organization of domain knowledge in terms of the structure of content associative tree. Then we examine some realistic aspects in the design of AVIS' structured knowledge base, such as the empirical method for acquiring AUTORED-oriented specific knowledge, the concepts of bi-polarized knowledge

expression and diagnostic weighting factor for dealing with the uncertainty in knowledge and data is described. Finally, the configuration of AVIS' knowledge base in terms of the sketch-details scheme.

### 2.3.1 Pattern-Based Knowledge Rarefaction: PADIKS Approach

Meaningful knowledge is concerned with expression, in some language or communicative medium that corresponds in some salient ways to the world or a state of the world (Brachman and Levesque, 1985). Knowledge rarefaction is a subject concerning the representation of the real world knowledge via a stylized version. The most intuitive and popular approach might be production rule shown in this format:

IF (premise) THEN (consequence).

This representation technique had dominated the paradigm of the expert systems in the past decade. However, a controversy associated with this scheme and also with other approaches of structured knowledge representation, (such as semantic network, frame, etc.) is that the knowledge expression and the rules that are used to derive inference from that knowledge need be written in terms of the many ways in which the knowledge may originally have appeared (Rich, 1983). This problem may be eliminated if we view the relationships between different knowledge items as a sort of knowledge parameter similar to the other knowledge variables, such as attribute,

predicate, etc. Therefore, all kinds of knowledge items can be expressed via uniform and abstracted patterns which, in return, form the common base for all kinds of knowledge processing. This pattern-based concept for knowledge rarefaction is the essence of PADIKS (Tou, 1985, Dai and Tou, 1988). For this approach to work, however, certain coherent properties must exist, such as semantical taxonomy which makes the relations between different knowledge items trackable. AVIS seems to meet this prerequisite very well. As we have mentioned in Chapter I, concerning error identification we can classify an error into 3 major categories: line, junction, and component, and further classify into more specific types, such as denotation line, junction line, component boundary line, and further into type-1 denotation line, type-2 denotation line, and so on.

Following the pattern-based approach, the domain knowledge of AVIS is represented in terms of three types of primitives: entities, attributes and relationships. Entities are types of information which are tied together by a set of relations to form a knowledge base. Attributes are characteristics of other knowledge entities. Attributes, when associated with a particular entity occurrence, may be valued or weighted. The characteristic attributes for each entity occurrence may be viewed as a pattern vector, which may be used as a basis for decision-making operations. The interrelationships between entities are called relationships.

The set of relationships virtually determines the overall structure of a knowledge domain. Alternatively speaking, attribute primitives are stylized symbols with absolute and invariant meanings. They are the vocabulary of the whole knowledge domain and are manipulated based on logical connectives. Entity primitives, representing certain generic concepts of interest, are abstracted knowledge with the associated realistic senses being a function of domain properties as well as problem solving strategies. An entity can be decomposed into components which are more elementary in the sense that each of them has a more homogeneous conceptual meaning. The decomposition of entities, in return, defines the third kind of primitives, relationships, denoting the epistemic structure of the domain. Referring to AVIS, we can define the problem domain as a defect space D containing various defect modes in the AUTORED generated data base. A defect mode, d, is an entity and it can be characterized by a conjunction of symptoms which are organized into a symptom vector, SV(d), shown in the following form:

$$SV(d)=[<S_1,v_1>, .., <S_i,v_i>]$$

where $S_i$ is symptom variable with symptom value $v_i$. Symptom variables are the attributes of AVIS domain. If we term the symptom vectors as descriptors (Cheng, 1983) associated with the defect modes, then a common descriptor might exist between two defect modes in case some of their symptom appearances are

the same. Therefore, the common descriptor forms a semantic link between these two defect modes, and determines the relationship primitives. Representing the domain knowledge of AVIS in terms of a small number of primitives, we virtually unify the diverse formation of knowledge to a common ground on which certain inference mechanisms (e.g., knowledge-directed pattern-recognition (Tou, 1985)) can be employed to process knowledge. The basic features associated with these three primitives are summarized in Table 2.1.

Table 2.1 The Basic Features of Three Kinds of Primitives.

| Entities | Attributes | Relationships |
|---|---|---|
| .generic concepts<br>.decomposable sense<br>.circumscribe knowledge domain<br>.relational-based | .invariant symbols<br>.absolute meaning<br>.encoding entities<br><br>.pattern-based | .subsumption relationships<br><br>.taxonomy-oriented |

A salient feature in this pattern-based approach is the usage of relationships, which dictates the adequacy and efficiency of knowledge representation. In semantic networks, the same notation of relationships is also employed, but in the rather simple format, such as is-a, a-kind-of. In AVIS, the canonical form is the content associative tree.

2.3.2 Knowledge Organization: Content Associative Tree

Content associative tree is a relational description about domain knowledge. It is in the form of hierarchical structure defined by semantic links (relationship primitives) with the addition of pertinent knowledge contents (in terms of entities and attributes) (Tou, 1985). The nodes of the content associative tree represent entities with each one in association with a set of attributes. The semantic links between entities determine the detail of the tree structure. The syntax of the content associative tree can be defined as follows:

Definition 2.1 The structure of content associative tree, $\pi$, containing p internal nodes and q terminal nodes, where $q \geq p+1$, $p \geq 0$, is defined recursively as follows:

1) If p=0, then it consists of a single terminal node,
2) If $p \geq 0$, it is a triple $(\pi_{p'}, t, \pi_{p''})$, where t is a distinguished external node of $\pi_{p'}$, called the root of $\pi_{p''}$, and $\pi_{p'}$, $\pi_{p''}$ are also in the hierarchy with p', p" internal nodes, respectively, and q', q" terminal nodes, respectively, where $q \geq p'+1$, $q'' \geq p''+1$, $p=p'+p''$, and $q=q'+q''$.

A salient feature of this structure lies on its proliferation aspect as shown in Figure 2.4. It allows the granularity of knowledge be properly controlled along the depth of the hierarchy. That means knowledge growth in this structure is

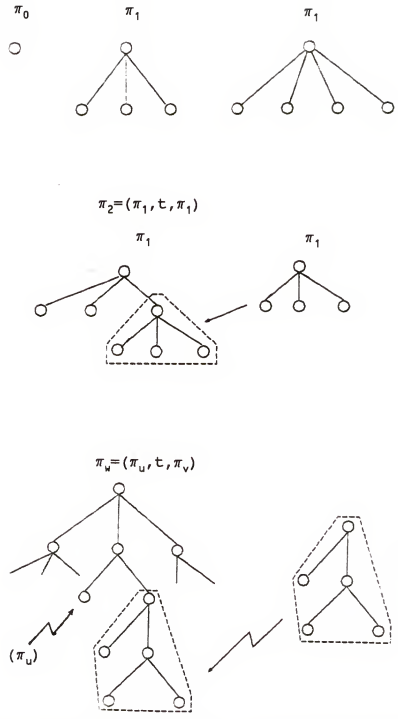



Figure 2.4 Structural Aspects of Knowledge Hierarchy, Where $\pi_2$ Is Assembled From Two $\pi_1$'s, And $\pi_w$ Is Assembled From $\pi_u$ and $\pi_v$.

easy, simply by expanding out the entities from the proper nodes. This structure, in fact, provides useful information to facilitate knowledge classification, new knowledge acquisition through insertion at the appropriate location in the hierarchy, identification of possible problems, and efficient seeking of appropriate knowledge in response to a query (Tou and Cheng, 1983b).

Semantic links. The realistic sense and the actual structure of the hierarchy just mentioned is realized by attaching to this hierarchy a group of relevant knowledge contents. In AVIS we organize the relational hierarchy based on a specialization scheme such that the most general concepts are placed at the top node of the tree, and the most specific items at the bottom of the hierarchy. The semantic links tie together the entity nodes and allow knowledge of various levels to be distributed throughout the hierarchy. Controlling the proliferation aspect of the relational hierarchy, the semantic links can be characterized as the node formulas with the following expression:

```
Node-Formula:
     Entity = [$ name]
     AC = [ancestor]
     DL = [descendant-list]
     KAL = [key-attribute-list]
     NAL = [new-attribute-list]
```

where AC specifies the ancestor of the entity, DL specifies the list of direct descendent entities, KAL denotes the unique features (attribute primitives not inherited from its parent node) associated with the entity, and NAL depicts the extra attribute primitives needed for breaking down the current entity into its descendants. The detailed formulation reflects the domain properties as well as the human expertise. They specify how the knowledge hierarchy will be assembled. Due to the content-oriented distribution scheme, the classification of entities (defect modes), may be overlapping as shown in the Venn representation in Figure 2.5. Hence, despite the tree-like appearance of the knowledge structure, the whole organization, in terms of defect mode classification, might be relationally associative, and thus it gets the name of content associative tree. The associated feature patterns in the node formulas may be viewed as the strategic information describing the centroid of the cluster of its sub-nodes.

Figure 2.6 shows a portion of the content associative tree of AVIS. The top-level node is Data-Completeness, which is followed by three entities: Superfluous-Line, Superfluous-Component, and Superfluous-Junction. Furthermore, under the node of superfluous line are included Denotation-Line, Noise-Created-Line, Component-Line, and so on. A basic constraint employed in AVIS is that we require the pattern descriptors associated with the direct descendants of a node be independent, i.e.,

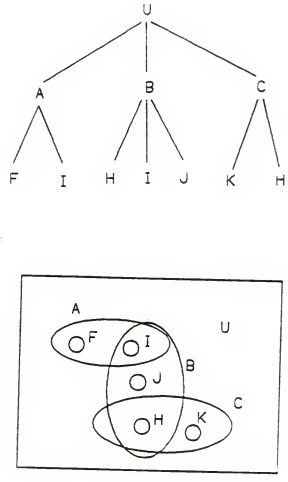



Figure 2.5 Category Overlapping in Content Associative Tree.

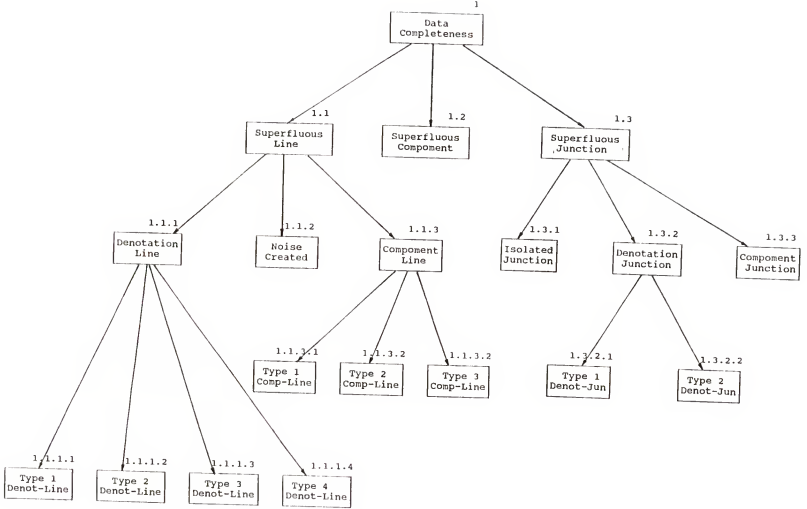



Figure 2.6 Domain Knowledge for Data Completeness Check Is Organized Into A Content Associative Tree.

$$r(e_{d1}) \cap r(e_{d2}) \cap \ldots \cap r(e_{dn}) = \phi$$

where $r(e_{dn})$ is the descriptor of the $n^{th}$ direct descendant, $e_{dn}$. Or in terms of KAL expression in node formula, we require that:

$$KAL_1 \cap KAL_2 \cap \ldots\ldots \cap KAL_n = \phi$$

where $KAL_n$ is the key-attribute-list of the $n^{th}$ descendant.

2.3.3 Utilization Aspect

A significant feature of this pattern-based approach is the representation of knowledge through various levels of details, which is consistent with our reasoning lines toward the usage of knowledge. This representation technique, generally speaking, provides a functional view of knowledge in terms of what it can represent and how it can be used. It is able to represent all the relevant knowledge that are needed in the domain, but also the additional information (i.e., structure) that can be used to focus the attention of the inference mechanism in the most promising directions. In other words, we are able to achieve, at one time, representational adequacy, inferential efficiency, as well as acquisition efficiency, three merits. These three aspects, in fact, are the common objectives pursued in the research of

structured knowledge representation (Rich, 1983). Nevertheless, the most significant advantage lies in the knowledge utilization aspect. The content-oriented knowledge organization coincides with the data-driven characteristic of abduction inference (Charniak and McDermott, 1985). For instance, in AVIS application, inference along the content associative tree might be to compare the symptoms observed from the data records with the descriptors associated with the nodes at the first level of the tree, then determine which node qualifies as a plausible solution (tentative solution state). Where a decision may be made by virtue of similarity measure between the observed symptoms and the feature profiles of the descriptors associated with the competing nodes. The node with highest score (closest match) may be selected as the basis for duplicating the same procedure with its descriptor as the directives for symptom extraction, and the direct subnodes as the new working hierarchy level. Figure 2.7 symbolically depicts the progressive aspect of the inference process. This inference process, clearly exposes the fact that abduction inference, in general, proceeds on the basis of observation (via data-driven scheme) with the inference path being adaptively adjusted following the appearance of evidence. In this regard an important aspect in the content associative tree is the hierarchical clustering of the domain entities created by the content-based classification strategy (Duda and Hart, 1973). Therefore, the data-driven feature

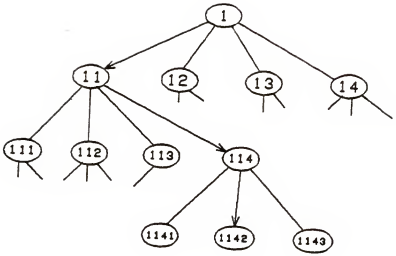



Figure 2.7 The Inference Process In Knowledge Sketch Starts From The Root Node. Then Gradually Moves Down To The Low Level Subnodes.

in abduction inference coincides very well with the content-oriented organization of the content associative tree. That means inference along the content associative tree can be natural and effective. Contrary to that, in category-based knowledge organization, where the whole domain is distributed into independent categorical groups, due to the incompatibility between knowledge structure and knowledge utilization process, certain degree of back-tracking during inference is inevitable. In AVIS, an inference control strategy, SICS, is designed to exploit this advantage and it will be further interpreted in Chapter III.

## 2.4 Design of AVIS' Structured Knowledge Base

The design of AVIS' knowledge base mainly makes use of the PADIKS concept just described. The following sections will discuss to a certain extent some technical aspects in the actual design. We begin with the knowledge acquisition process for eliciting AUTORED-oriented domain knowledge.

### 2.4.1 AUTORED Experience Acquisition

As we have briefly introduced at the beginning of this chapter, the domain knowledge of AVIS comes from three directions: design conventions, circuit theories, and AUTORED experience. The general verification knowledge from the first two categories can be assessed from textbooks, design manuals, or through consultation with experts. However, for the last

source, it is specific knowledge, case-oriented, and has to come from AUTORED itself. This set of knowledge, because of being domain-specific, tends to be the most important and effective for the verification operation. A trouble in this concern is the overwhelming complexity of AUTORED's picture taking mechanism which, as shown in Figure 2.8, very much prohibits the construction of a proper model for knowing the occurrence of data flaws in the generated CAD data base. To alleviate this difficulty, we take an empirical strategy, and Figure 2.9 presents the flow-chart of the knowledge acquisition process. We first manually convert the AUTORED captured data records in the CAD database into the corresponding schematic, denoted M.G.S. (shorted for machine generated schematic), which, in return, is compared with the original schematic fed to AUTORED. The differences between these two schematics are identified and analyzed. Those discrepancies representing viable clues and statistics of AUTORED committed mistakes, are described in natural language as the preliminary resources of domain knowledge AVIS needs. This raw verification knowledge is further mechanized following the process shown in Figure 2.10, where, in the human channel, the defect descriptions in high level natural language are translated in the machine channel into the abstracted format containing specialized feature variables termed diagnostic description language (DDL). The defect description in DDL is further arranged into the pattern vector

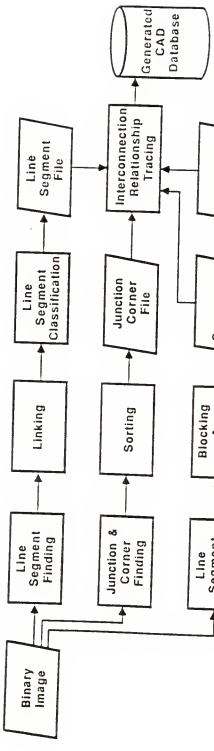
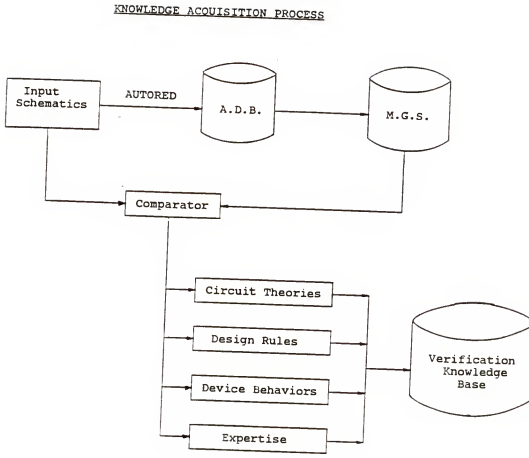



Figure 2.8 The Picture Taking Process of AUTORED.




A.D.B.: AUTORED Captured Data Base.

M.G.S.: Machine Generated Schematics.

Figure 2.9 The Knowledge Acquisition Process in AVIS.

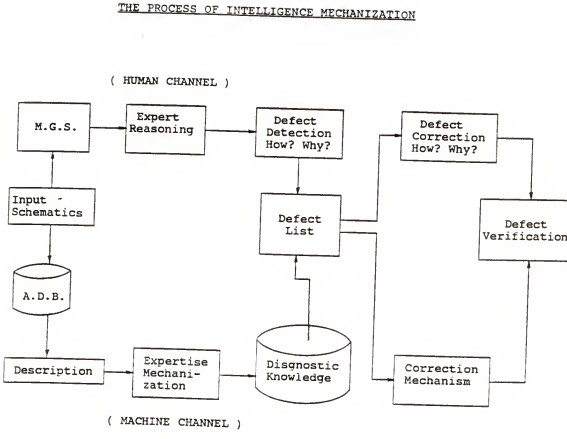



Figure 2.10 The Process of Intelligence Mechanization.

format (descriptor) which is the ultimate expression of knowledge used by the system. In general, a defect instance might contain several characteristic aspects. If each characteristic aspect is termed a descriptor, then a defect instance might be represented by a disjunction of descriptors as:

$$\text{Feature Profile of } d_i = r_1(d_i) \cup r_2(d_i) \cup \ldots \cup r_q(d_i)$$

Each descriptor is termed as a variation of the defect mode. The following example about SUPERFLUOUS JUNCTION summarizes the whole translation aspects just mentioned:

Example: superfluous junction

Human channel: (natural language)

"Superfluous junction occurs as a junction symbol locating outside any component blocks. It has erroneous junction type indication with no or only one line connection."

Machine channel: (DDL)

```
(JLOC, NICB):=> junction location is not inside component
                block
(JUN-TYPE-CHECK, ERRONEOUS):=> junction type is erroneous
(LINE12-STATUS, MISSING):=> line-segment-12 is missing
(LINE12-STATUS, EXIST):=> line-segment-12 is existing
(LINE34-STATUS, MISSING):=> line-segment-34 is missing
(LINE34-STATUS, EXIST):=> line-segment-34 is existing
```

Descriptors of Superfluous Junction:==>

<(LINE12-STATUS, MISSING), (LINE34-STATUS, EXIST),
(JLOC, NICB), (JUN-TYPE-CHECK, ERRONEOUS)>

U <(LINE34-STATUS, MISSING), (LINE12-STATUS, EXIST),
(JLOC, NICB), (JUN-TYPE-CHECK, ERRONEOUS)>

U <(LINE12-STATUS, MISSING), (LINE34-STATUS, MISSING),
(JLOC, NICB), (JUN-TYPE-CHECK, ERRONEOUS)>

Once the flaws in M.G.S. have been identified, therapies (certain appropriate actions) can be planned to make corrections, and those planned actions embody AVIS' validation mechanism. Following the same token, the correction plans in the human channel expressing our strategies for making compensation are translated in the machine channel into the corresponding procedures realizing those actions as also shown in Figure 2.10. The process is repeated on several schematics in order to collect a more complete set of AUTORED-oriented verification knowledge. Depending on their properties, the acquired knowledge is categorized into different groups with each one termed a Defect Information Unit (DIU) as the ready source of verification knowledge. Totally four DIUs (line-DIU (L-DIU), junction-DIU (J-DIU), component-DIU (C-DIU)) and topology-DIU (T-DIU) are obtained. The general knowledge acquired from the other two sources, design conventions and circuit theories, are also encoded by the same fashion. Here, a production rule is considered as a kind of pattern, but the

"consequence" part of the rule only signifies "certain indications" rather than "firing certain actions" (Tou, 1985). An issue different from above is the domain knowledge for interconnection verification, where the problem domain has little sense of classification. It is acquired based on different models and methodologies and it will be explored in Chapter V.

### 2.4.2 Knowledge Fine-Tuning

The knowledge descriptors collected in each DIU are fine-tuned for two purposes: (1) to cope with the imprecision of knowledge, and (2) to create the special scheme of bi-polarized knowledge expression.

#### Diagnostic weighting factors (DWF): The uncertainty in knowledge

Knowledge base is a repository of human knowledge, and since much of human knowledge is imprecise in nature, therefore, how to represent the reliability of each individual knowledge expression becomes a fundamental issue (Rich, 1983). Diagnostic weighting factor (DWF) is devised in response to this concern. The DWF settings represent the imprecision of knowledge contents via a numerical means. Generally speaking the settings of DWF contrast the significance difference of attributes in characterizing a specific entity. The determination of the DWF settings, unlike the attribute value

which can readily be determined from observation or other sources, is knowledge intensive and is dependent upon the designer's expertise and experience. It may be a number subjectively decided from the expert's opinion, or obtained from certain formalisms, such as conditional probability (Chang and Tou, 1983). For AVIS, the setting of DWF is determined by the frequency of occurrence of the attribute which appears in the descriptor characterizing the defect modes. The value so obtained has the sense quite similar to the conditional probability (probability of attribute conditioned on defect mode) but they are not explicitly advertised since no rigorous method has been used to justify its probability meaning. In practice, these values are further adjusted on the basis of run-time performance. The usage of DWFs regarding approximate reasoning will be interpreted in Chapter IV.

Bi-polarized knowledge expression. Bi-polarized knowledge expression is a special measure arranged in AVIS' knowledge base. The purpose of this augmentation is to extend the language dimension about knowledge expression, and most importantly, make the decision process converge quickly. The premise of this technique is to expand the descriptors associated with defect modes into two extremes: supportive and adverse. The feature profile of defect mode, $d_i$, based on this scheme can be expressed as:

Feature Profile of $d_i$ ==> $[r_1(d_i) \cup r_2(d_i) \cup \ldots \cup r_q(d_i)] \cup [\neg r_{1'}(d_i) \cup \neg r_{2'}(d_i) \cup \ldots \cup \neg r_{q'}(d_i)]$
= [SUP-DSPT] U [ADV-DSPT]

with $r_q(d_i) = (w_1, w_2, \ldots, w_p)$
$\neg r_q(d_i) = (w_{\neg 1}, w_{\neg 2}, \ldots, w_{\neg p})$

Where $r_q(d_i)$, $\neg r_q(d_i)$, are the supportive and adverse descriptor, respectively, $w_p$, $w_{\neg 1}$, are the supportive and adverse attribute, respectively. The supportive attribute, $w_p$, is defined as the attribute with its appearance contributing to the confirmation of the truth of the associated entity (defect mode). On the contrary, the adverse attribute, $w_{\neg p}$, is defined as the factor which counts against the truth of the associated entity (defect mode). As an illustration, in AVIS' problem domain the denotation line symbol usually appears as a short line segment with irregular connections. It might be close but never resides within a component block. This statement can be abstracted into the following descriptor:

```
DENOT-LINE:==>
    SUP-DSPT = <(LINETYPE, INT),(LLOC, CCB),
                (LCON-STATUS, ISOLATE)>
    ADV-DSPT = <(LLOC, ICB)>
```

where SUP-DSPT, ADV-DSPT are the supportive and adverse descriptor, respectively. The ADV-DSPT specifies that line location (LLOC) is inside the component block (ICB). As a consequence, when we encounter a line segment with location inside a component block, we immediately have a strong doubt about the possibility for the line segment to be a denotation line symbol. By this way, we combine evidence from a broader basis. If the inference logic doesn't involve ADV-DSPT, even if the attribute-value-pair <LLOC, ICB> has been observed, based on similarity concept, it might be viewed as irrelevant, since it makes no contribution to the truth of the SUP-DSPT, nor to the associated entity. However, with the inclusion of the adverse descriptor, we can distinguish irrelevant evidence from adverse evidence clearly. Irrelevant evidence can be defined as the evidence neither supportive nor adverse. The concept of bi-polarized knowledge expression can be alternatively applied to different knowledge levels including attribute, descriptor, and defect mode. Its actual representation in terms of evidence space (Rollinger, 1983) as well as the manipulation for pooling contradictory information will be the key subjects of Chapter IV.

Generally speaking, the set of adverse descriptors in AVIS is obtained based on our judgement in addition to the general definitions in AVIS. For instance, the adverse descriptor associated with component-boundary-line is decided as <LLOC, NICB> (line location is not inside component block),

since by definition a component-boundary-line can appear only inside a component block. The set of adverse descriptors is stored in a file called consultation module, which, as the name implies, is only for verification consultation without extending into the knowledge hierarchy.

### 2.4.3 Configuration of AVIS Structured Knowledge Base

Following the blueprint of the pattern-based knowledge representation technique as described earlier, the configuration of AVIS' structured knowledge base storing the set of acquired and fine-tuned domain knowledge, contains three major parts: knowledge sketch, knowledge details, and context generator.

Knowledge sketch. A content associative tree with no free variables in the node formulas is defined as knowledge sketch (Tou and Cheng, 1983b). It encompasses the outlines of the whole knowledge domain. The knowledge sketch is represented by hierarchical entity-attribute-value relations. This structural knowledge is semantically categorized into an associative tree. Each node of the relational hierarchy represents an entity (defect mode) associated with a set of attributes and values (symptoms) which form the information patterns describing the knowledge needed for verification. In the current AVIS design, the domain knowledge is organized into two knowledge sketches targeted at two verification

usages: data completeness check and data consistence check, respectively. Totally 7 modes of line defects, 15 modes of junction defects, and 8 modes of component defects are categorized and included in the knowledge sketches. Figure 2.11 depicts the category structure in the knowledge sketch for data consistency check.

Regarding to the utilization of knowledge, each associated entity is a task state of the domain universe. Hence, the knowledge sketch becomes the know-how set concerning the ways of utilizing those task states for problem solving. The attributes associated with the entities are the strategic information and directives for inference. In AVIS, pattern recognition principles are used in the design of knowledge seeking strategies for making accurate diagnosis and positive identification. The exclusive discussion will be given in Chapter III.

Knowledge details. As mentioned early, knowledge entities are types of information which are tied together by a set of relationships to form a structured knowledge base. The unit of knowledge details realizes this concept. It is composed of a collection of tables and files concerning the entity code names, the semantic links, and the feature patterns, and others in supporting the functioning of knowledge sketch. The key design aspect lies in data retrieval, transfer, extension, management, etc. In the

PART OF CATEGORY STRUCTURE FOR DATA CONSISTENCY VERIFICATION

| CATEGORY CODE | DEFECT MODE NAME |
|---|---|
| 3 | Data Consistency |
| 3.1 | Partially Recognized Component |
| 3.1.1 | Type |
| 3.1.1.1 | JFET Transistor |
| 3.1.1.1.1 | JFET Terminal Type Unknown |
| 3.1.1.1.2 | JFET Terminal Type Known |
| 3.1.1.2 | BJT Transistor |
| 3.1.1.2.1 | BJT Terminal Type Unknown |
| 3.1.1.2.2 | BJT Terminal Type Known |
| 3.1.1.3 | MOSFET Transistor |
| 3.1.1.3.1 | Terminal Type Unknown |
| 3.1.1.3.2 | Terminal Type Known |
| 3.1.1.4 | Port |
| 3.1.1.4.1 | Input Port |
| 3.1.1.4.2 | Output Port |
| 3.1.1.4.3 | Ground |
| 3.2 | Junction Type Error |
| 3.2.1 | Group-1 Junction Type Error |
| 3.2.1.1 | Type-1 Junction Type Error |
| 3.2.1.2 | Type-2 Junction Type Error |
| 3.2.1.3 | Type-3 Junction Type Error |
| 3.2.2 | Group-2 Junction Type Error |
| 3.2.2.1 | Type-4 Junction Type Error |
| 3.2.2.2 | Type-5 Junction Type Error |
| 3.2.2.3 | Type-6 Junction Type Error |
| 3.2.2.4 | Type-7 Junction Type Error |
| 3.2.3 | Group-3 Junction Type Error |
| 3.2.3.1 | Type-8 Junction Type Error |
| 3.2.3.2 | Type-9 Junction Type Error |

Figure 2.11 Part of the Category Structure
For Data Consistency Verification

current version, the knowledge details are arranged to host 5 essential elements including knowledge hierarchy, attribute set, entity-attributes relationships, attribute-context-generator relationships, and diagnosis-treatment relationships. The knowledge hierarchy concerns the distribution of entities and provides a comprehensive coverage of verification outlines over the problem domain. The attribute set contains those attributes significant in diagnosis. The entity-attribute relationships materially link together the entities with the attribute set, therefore, they make the knowledge sketch meaningful. Attribute-context-generator expresses the correspondence between attributes and context generator which are certain analytic routines capable for extracting relevant diagnostic context from the background data. The data structure for this unit consists of hierarchical node index table (NITA), entity dictionary (ED), attribute dictionary (AD), entity-attribute table (EAT), attribute-context-generator table (ACGT), validation-plan dictionary (VPD), and defect-validation-plan table (AVPT). The NITA plays a significant role in this unit. It represents the domain knowledge sketch base in an associative tree structure, the analogue to the knowledge sketch concept as described in the last section. Figure 2.12 summarizes the information flow in this unit. The design in these data files is to achieve three functional objectives:

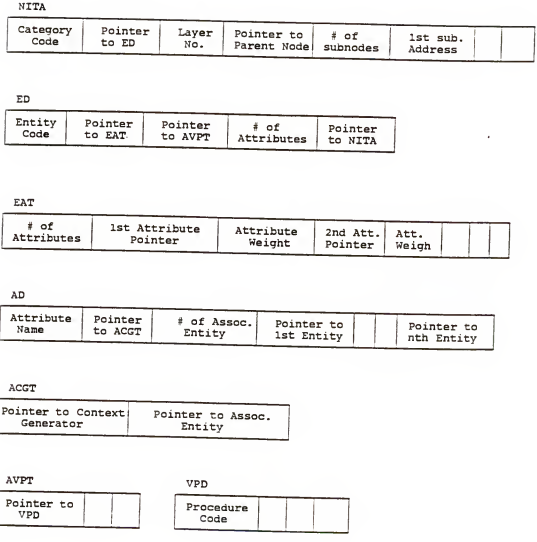



Figure 2.12 AVIS Data Structure and Information Flow

(1) an entity is retrieved on the basis of an associated attribute.

(2) an attribute is retrieved on the basis of associated entity.

(3) cross-traverse within the knowledge sketch is allowed from root level to leaf level or vice versa.

As will be further interpreted in the next chapter, these three functions are essential for the integral analysis, differential analysis, two major components in our inference control strategy. They are accomplished in conjunction with the inverted file technique (Wirth, 1976). The APRIKS system (Tou and Cheng, 1983b) also contributes the design blueprint of this portion.

Context generator. In conventional expert systems, attributes for symbolic reasoning are generated through a question-answer process in the man-machine interaction phase. The system, then, utilizes the contexts containing the user provided attribute information for rule selection and the related knowledge activities. In AVIS, we define the contexts as a set of descriptions or symptoms of a mis-recognized circuit element which needs to be identified. That is contexts are the attributes actually extracted from the data record and used for inference. As an automatic system, AVIS generates the contexts based on its self-sensing capability without resorting to man-machine interaction, and the context

generator realizes such capability. Context generator is a collection of analytic programs capable of abstracting general low level data into symbolized attribute primitives. Those extracted attributes termed diagnostic contexts denote the relevant features or manifestations associated with the flaws in the CAD data base. Diagnostic contexts are the ultimate evidential information on which AVIS relies for verification operations. In the current version of AVIS, the context generator contains four groups of program routines including:

(1) Line Group Specializing on line domain, this group consists of 5 program routines responsible for analyzing the properties and features of the data records inside the line data files. Totally 11 line features are taken into account; a portion of them is shown in Figure 2.13.

(2) Junction Group This group, being in charge of junction features extraction, consists of 6 program routines which access 20 junction features. Some of them are shown in Figure 2.14

(3) Component Group Specializing on component feature extraction, at present, there are 4 program routines in this group and 13 different component features are taken into account. Figure 2.15 depicts some of the component features utilized in AVIS.

(4) Topology Group The purpose of this group is to interpret the connectivity of the component symbols in the captured CAD data base. It is also used for interconnection

CONTEXT CATEGORY AND CONVERSION CODE: LINE GROUP

| ATTRIBUTE | VALUE | GENERATED CODE | CONVERTED CODE |
|---|---|---|---|
| LINETYPE | INT | 1 | <101, 1> |
| | EXT | 0 | <101, 2> |
| LENGTH | SHORT | 1 | <102, 1> |
| | LONG | 0 | <102, 2> |
| L-ORIENT | HORIZON | 0 | <103, 1> |
| | VERTIC | 1 | <103, 2> |
| ENDCON1 | CMP# | 1 | <104, 1> |
| | JUN# | 2 | <104, 2> |
| | LINE# | 3 | <104, 3> |
| | nill | 0 | <104, 4> |
| ENDCON2 | CMP# | 1 | <105, 1> |
| | JUN# | 2 | <105, 2> |
| | LINE# | 3 | <105, 3> |
| | nill | 0 | <105, 4> |
| LLOC | ICB | 1 | <106, 1> |
| | NICB | 0 | <106, 2> |
| | CCB | 2 | <106, 3> |
| | IDNT | 4 | <106, 4> |
| | NIDNT | 3 | <106, 5> |
| | | | |

Figure 2.13 Part of Line Contexts Used in AVIS.

CONTEXT CATEGORY AND CONVERSION CODE: JUNCTION GROUP

| ATTRIBUTE | VALUE | GENERATED CODE | CONVERTED CODE |
|---|---|---|---|
| JLOC--STATUS | CLUSTER | 0 | <201, 1> |
| | SPARSE | 1 | <201, 2> |
| CLUSTER--JUN | (1 == 7) | (1 == 7) | <202, 1==7> |
| | NILL | 0 | <202, 8> |
| COMMON-BASE | NON-EXIST | 1 | <203, 1> |
| | LINE12 | 2 | <203, 2> |
| | LINE34 | 3 | <203, 3> |
| LINE12--STATUS | EXIST | 1 | <204, 1> |
| | MISSING | 2 | <204, 2> |
| LINE12-LINETYPE | INT | 1 | <205, 1> |
| | EXT | 0 | <205, 2> |
| LINE12--LLOC | ICB | 1 | <206, 1> |
| | NICB | 0 | <206, 2> |
| | CCB | 2 | <206, 3> |
| | IDNT | 4 | <206, 4> |
| | NIDNT | 3 | <206, 5> |
| | | | |

Figure 2.14 Part of Junction Context Used in AVIS.

CONTEXT CATEGORY AND CONVERSION CODE: COMPONENT GROUP

| ATTRIBUTE | VALUE | GENERATED CODE | CONVERTED CODE |
|---|---|---|---|
| NAME | KNOWN | * | <301, 1> |
| | UNKNOWN | 8277,8267,8275 | <301, 2> |
| PIN | (1 == 3) | (1 == 3) | <302, 1==3> |
| OVERLAP--COMP--NAME | R | 8261 | <303, 1> |
| | C | 8279 | <303, 2> |
| | D | 8281 | <303, 3> |
| | BJT | 8258 | <303, 4> |
| | JFET | 8266 | <303, 5> |
| | INPUT | 8265 | <303, 6> |
| | OUTPUT | 8276 | <303, 7> |
| | UNKNOWN | 8267 | <303, 8> |
| | NOISE | 8277 | <303, 9> |
| | TP | 8275 | <303, 10> |
| | VOLTAGE | 8272 | <303, 11> |
| | GND | 8263 | <303, 12> |
| | DNT | # | <303, 13> |
| | NILL | 0 | <303, 14> |
| CLOC | MID-LEFT | 1 | <304, 1> |
| | MID-RIGHT | 2 | <304, 2> |
| | | | |

Figure 2.15 Part of Component Contexts Used in AVIS.

verification and comfort visualization during man-machine interaction. Right now, 16 program routines are included. These programs are under the direct control of AVIS' inference mechanism (Chapter III).

The configuration. The ultimate configuration of AVIS' knowledge base is shown in Figure 2.16. As is evident from the above discussion, knowledge representation via knowledge sketch is syntactic, but also semantic, and takes into account the uniform and structural aspects in the whole expression. It is declarative but also operational, since the whole set of entities (task states) has been explicitly represented but also closely linked, and thus the whole knowledge domain becomes a working unit. With the inclusion of the context generator, the system is capable of integrating the traditional problem solving capabilities of expert systems with the precision of numerical computing. Moreover, since attributes (abstracted knowledge patterns) are the sole communication medium among these three knowledge components, the functional coordination is easy and systematic. In the next chapter, we move on to the next subject regarding the utilization of the organized knowledge base.

## 2.5 Summary

The research area of knowledge representation has a long, complex, and diverse history. Much has been written of the

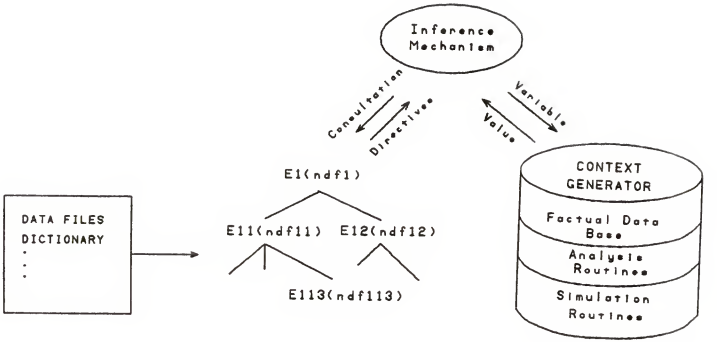



Figure 2.16 The Configuration of AVIS' Structured Knowledge Base.

discord and disagreements (Brachman and Levesque, 1985). In designing AVIS' knowledge base, instead of following the tradition of representing knowledge in terms of rules, clauses, or other formal languages, our alternative is pattern-based and stresses the importance of knowledge structure, which has been recognized as essential for effective inference (Chandrasekaran, 1984). We represent the domain knowledge in terms of three primitives; entities, attributes, and relationships, and further organize the stylized notations into a structure of content associative tree. Thus, we can achieve some significant advantages including representational adequacy, inferential efficiency, and acquisition efficiency, three major objectives pursued in knowledge engineering research.

AVIS' domain knowledge is extracted from three sources: circuit theories, design conventions, and the AUTORED experience. The last source is domain specific and an empirical method for this purpose is presented. Acquired domain knowledge is further fine-tuned to cover (1) the uncertainty in knowledge, and (2) the adverse descriptors for accerating decision making. Regarding the pattern-based knowledge representation scheme, we propose that the configuration of AVIS' knowledge base be composed of three parts; knowledge sketch--characterizing the outlines of the domain knowledge, knowledge details--consisting of the data files and dictionaries of the knowledge base, and context

generator--including a collection of procedures for extracting the diagnostic contexts from the problem domain. This architecture is able to integrate into the system the versatility of symbolic reasoning with the accuracy of numerical computing. Some realistic design aspects are also elaborated.

A general discussion about the generation of knowledge structure is included in the APPENDIX, where we categorize knowledge domains into two types: well-organized, and semi-organized. For the latter case, we propose that the knowledge domain be modeled by an N-cube, and the knowledge structure be generated based on the technique of N-cube partition.

# CHAPTER III
# STRUCTURED INFERENCE AND SICS INFERENCE STRATEGY

## 3.1 Introduction

As stated in Chapter II, knowledge representation and organization are concerned with the encoding of knowledge and information into stylized symbols and data structure, and AVIS' domain knowledge is organized into a sketch-details scheme. The main subject of this chapter is knowledge utilization regarding the practical manipulation of the specialized data and structure to make intelligent inferences. Generally speaking, the formalism of content associative tree, the knowledge structure of AVIS, is excerpted from the principles and technologies of pattern recognition plus information retrieval. Pattern recognition is good at inference and consultation, while information retrieval is famed for automated knowledge transfer and utilization (Tou, 1985). Together, they form the basis of structured knowledge representation, also delineate the guidelines of inferences. AVIS' knowledge sketch is organized into the form that the root node represents a generic classification describing the overall coverage of defect modes, and the leaf nodes represent the specific form a comprehensive basis such that any defect mode which might be classified under the root may be correctly

classified under one or more leaf nodes. A simple inference process might be to compare the diagnostic contexts extracted from the data record with the descriptors associated with the nodes at the first level in the knowledge hierarchy, then determine which node qualifies as the plausible solution, where a decision might be made by virtue of similarity measure between the diagnostic contexts and the feature profile of the descriptor of each competing node. The node with highest score (closest match) can be selected as the basis for duplicating the same procedure with its direct subnodes as the new working hierarchy level. This process might iterate until a leaf node is selected. From this standpoint, knowledge seeking in the knowledge sketch has the sense of solution searching in the space of possibility, and the control metaphor in this simple reasoning process is depth first. The inference strategy Establish/Refine in MAX (Chandrasekaran, Sanjay, 1983) for making positive identification of disease is a variation of the depth first strategy.

Solution search along the typical tree structure has been studied extensively in the AI (artificial intelligence) community (but with different concerns, such as route finding, game playing). The well known procedures include depth-first search, hill-climbing, breadth-first search, and beam search. When the cost of traversing a path is of primary importance, there are more complicated procedures such as branch and bound, and dynamic programming. A common feature associated

with those search algorithms is the employment of heuristic pruning of alternatives, which makes explosive problems computationally feasible, but runs into the risk of losing the optimal solution. Back-tracking to a certain extent in those algorithms is inevitable. Some assumptions might also be imposed in realistic applications. For instance in Establish/Refine, it is assumed that diagnosis can always proceed from the general to the specific in a serial manner, with one and only one refinement of an established hypothesis can be selected, without guessing, back-tracking, or getting stuck. However, how many interesting domains satisfying these limiting assumptions is a point of controversy (Szolovits, 1985).

A novel approach, termed interactive negotiation strategy (Chang and Tou, 1984), makes use k-mean clustering technique (Tou and Gonzalez, 1974) as the mechanism of inference. This approach, equipped with the a priori information about the disorder number (provided by the user based on his/her expertise or simple guess), delivers a list of candidates with attached significance scores (similarity measures) and other relevant information. The candidate list constructively inspires the user's deep thought, therefore, the user can either adjust the setting of disorder number and repeat the negotiation process, or simply accept the system's suggestion and halt the consultation. This technique (k-mean) has the flavor of beam search with the beam-width being specified by

the user, who, in fact, takes the ultimate responsibility and judgement of diagnosis.

In the following we illustrate an alternative, termed SICS, short for squeeze inference control strategy. This inference control is specialized on the structure of content associative tree. It is a kind of beam search with the beam-width, however, adaptively adjusted based on diagnostic contexts. This adaptivity is beneficial in AVIS since AVIS behaves like an automatic system; thus, negotiation with the user during inference is generally impossible. Moreover, the solution generated by SICS, conditioned on the knowledge sketch and the observability of attributes, can be proved to be complete and concise, and thus, back-tracking, in theory, is not needed.

## 3.2 Control Principles In SICS

The inference control of SICS consists of two mechanisms: integral analysis and differential diagnosis. Referring to the diagnostic model depicted in Chapter I, their functions can be expressed as the following:

Integral Analysis:

$$\text{SKB} * \text{DCNTX} \Longrightarrow \text{IA} \Longrightarrow \{ \hat{E}_j \}; \quad \hat{E}_j \in (E_0, E_1, \ldots, E_i, \ldots, E_n)$$

Differential Diagnosis:

$$SKB * DCNTX * \{ \hat{E}_j \} \Longrightarrow DD \Longrightarrow \hat{E}_k^* \; ; \quad \hat{E}_k^* \in (E_0, E_1, \ldots, E_i, \ldots, E_n)$$

where

SKB = structured knowledge base

DCNTX = diagnostic contexts

IA = integral analysis strategy

DD = differential diagnostic strategy

$E_i$= $i^{th}$ (pre-enumerated) error mode (i=0,1,...,n)

$\hat{E}$ = plausible error mode

$\hat{E}^*$ = diagnosed error mode

Technically speaking, the integral analysis, IA, is designated to the process of inducing a set of plausible solution states, $\{E_j\}$, based on knowledge base and the observed evidence. In medical diagnosis, a doctor might think of a couple of diseases based on the symptoms observed in the patient. For the case of AVIS, the system induces a set of plausible defect modes based on the extracted diagnostic contexts. Differential diagnosis, DD, on the other hand, is trying to discriminate from this set of plausible solution states, $\{E_j\}$, the one, $\hat{E}_k^*$, most favorable, for instance identifying the actual disease of the patient, or making positive identification of defect modes. In order to narrow

options, the differential diagnosis usually requests more diagnostic observations to contrast out the differences among the competing solution states. Due to the newly explored evidence, more plausible solution states might be raised from the integral analysis, and they, thus, stimulate a deeper differential diagnosis. The dynamics of SICS basically comes from the alternatives of these two analyses. However, the essence in SICS' integral analysis is its capability of clustering the upper bound of the potential defect modes (which virtually eliminates the necessity of back tracking). Furthermore, in the differential analysis (in conjunction with a termination check routine), SICS manages to find the best upper bound as well as the best lower bound of the plausible defect modes. Hence the system can terminate the inference process when these two bounds become equal. Equality exists when the components of the problem domain are discrete and finite. Figure 3.1 illustrates the overall idea.

The upper bound concept, as will be further interpreted, is particularly useful in multi-faults diagnosis which, due to the rapid diversity of choice of plausible solutions as well as their combination, virtually cripples all kinds of search algorithms with sequential control character. However, incorporating the notion of content associative tree, the upper bound concept proceeds a categorical reasoning with a hierarchical clustering sense, therefore, it eliminates the sequential and exhaustive testing a large number of active

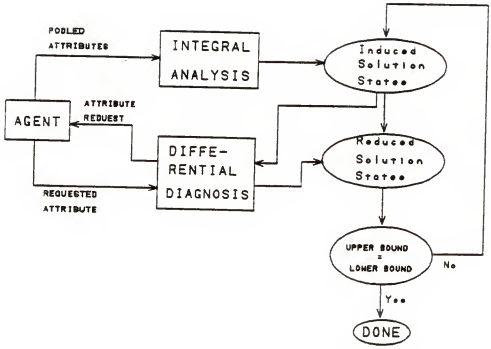



Figure 3.1 SICS Inference Mechanism Contains Integral Analysis and Differential Diagnosis.

hypotheses. In the problems of single-fault diagnosis, SICS is also convergent with the beam-width narrowing to the minimum (e.g., one). The theoretic groundwork of SICS is derived from the rough sets theory.

### 3.3 Mathematical Preliminaries: Rough Sets Theory

The following review about rough sets theory is excerpted from (Pawlak, 1982). Only those related to our research interest are introduced.

Let U be a certain set called the universe, and R be an equivalence relation (indiscernible relation in general) on U. The pair A=(U,R) is called an approximation space. Equivalence classes of the relation R are called elementary sets (or atoms) in A. The empty set is assumed to be elementary in every A. Every finite union of elementary sets in A is called a composed set in A, and the family of all composed sets in A is denoted as Com(A).

Let X be a certain subset of U. The least composed set in A containing X is called the best upper approximation of X in A, in symbols $\overline{\text{Apr}}_A(X)$. The greatest composed set in A contained in X is called the best lower approximation of X in A, in symbols $\underline{\text{Apr}}_A(X)$. The set $\text{Bnd}_A(X)=\overline{\text{Apr}}_A(X)-\underline{\text{Apr}}_A(X)$ is called the boundary of X in A, and sets $\underline{\text{Edg}}_A(X)=X-\underline{\text{Apr}}_A(X)$ and $\overline{\text{Edg}}_A(X)=\overline{\text{APR}}_A(X)-X$ are referred to as an internal and an external edge of X in A, respectively. Obviously, we have

$\overline{Bnd_A}(X)=\overline{Edg_A}(X)\cap\underline{Edg_A}(X)$. Figure 3.2 shows the notion of an upper and lower approximation in a two dimensional approximation space consisting of a rectangle partitioned into elementary squares.

There are two membership functions, strong and weak membership function, respectively, affiliated with an approximate space. Strong membership function, denoted $\underset{\sim}{\epsilon}_A$, and weak membership function, denoted $\bar{\epsilon}_A$, are defined as follows:

$$x \underset{\sim}{\epsilon}_A X \quad \text{iff} \quad x \in \underline{Apr_A}(X)$$
$$x \bar{\epsilon}_A X \quad \text{iff} \quad x \in \overline{Apr_A}(X)$$

If $x \underset{\sim}{\epsilon}_A X$, we say that "x surely belongs to X in A," while $x \bar{\epsilon}_A X$ is to mean that "x possibly belongs to X in A."

If $\underline{Apr_A}(X)=\overline{Apr_A}(X)$ for every $X\epsilon U$, then A=(U,R) will be called a discrete approximation space.

Other aspects about the rough sets theory, such as rough equality, rough inclusions, etc., can be found in (Pawlak, 1982).

## 3.4 Knowledge Domain: View From Rough Set Theory

As elaborated in Chapter II, the domain knowledge of AVIS is represented following the pattern-based approach. Defect modes are characterized by their associated descriptors, and each descriptor is characterized by a collection of

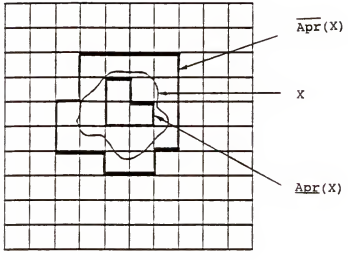



Figure 3.2 The Basic Concepts of Rough Sets Theory.

attributes. These knowledge items are further organized into a structure of a content associative tree, termed knowledge sketch. Conceptually, AVIS domain knowledge can be expressed (modeled) as the quadruplet:

$$\Phi = < D, \Omega, \text{Å}, \Psi >$$

where

D is the defect space consisting a complete set of defect modes happening in the data base,

$\Phi$ denotes the domain knowledge representing the strategic information known a priori to AVIS for verification,

$\Omega$ is the set of attributes denoting the symptom features associated with all kinds of defect modes,

Å= {$\alpha$}. $\alpha$ is defined as the characteristic mapping from the defect mode, d, to the associated attribute set, i.e.,

$$\alpha(d) = <w_1, w_2, \ldots., w_p>$$
$$= \text{descriptor of } d.$$

Å is the set of characteristic mappings from D to the power set of $\Omega$,

$\Psi$ is AVIS' knowledge sketch.

Following the premise of the rough sets theory, if we interpret the characteristic mappings Å over D in such a way that

$$d_i \sim d_j \quad \text{iff} \quad \alpha(d_i)=\alpha(d_j)$$

then obviously Å is an equivalence relation, and Ü=(D,Å) is

the approximation space induced by the defect space and the characteristic mappings which describe the knowledge domain. In accordance with the notations used in rough set theory, in the following, we term the defect modes residing at the leaf nodes of the knowledge sketch atoms, and the entity associated with the intermediate nodes hypotheses. In general, a hypothesis, $h_i$, is a composed set of atoms or other hypotheses, and the descriptor associated with the entity represents the feature profile of the hypothesis. When a hypothesis is at a leaf node, it is an elementary set containing a single atom. The relations between AVIS' knowledge domain and the corresponding approximate space are tabulated as follows.

Table 3.1 The Correspondence Between Knowledge Domain and Approximate Space.

| Knowledge Domain | Approximate Space |
|---|---|
| . Defect Space D | ==> Universe |
| . Mapping relation $\alpha$ | ==> Equivalence Relation |
| . Knowledge Domain | ==> Approximate Space |
| . Primitive Defect Mode | ==> Atom: Elementary Set |
| . Entity | ==> Hypothesis: Composed Set |

## 3.5 Inference Problem Formulation

Based on the approximate space Ü just defined, the problem of inference in AVIS' knowledge base (i.e., knowledge sketch) can be formulated as a 6-tuple $<\mathbb{T}, H, A, \Omega, DCNTX, \mathring{A}>$: where

$\mathbb{T}$ is the knowledge sketch of AVIS. Here we assume the hierarchy proliferation is based on specialization scheme, i.e., lower level nodes have descriptors with more attributes than their upper level nodes.

$H = \{h_1, h_2, .., h_i, .. , h_n\}$ is the finite set of hypotheses associated with the nodes in the knowledge sketch.

$A = \{a_1, a_2, .., a_i, .. , a_n\} \subseteq H$ is the finite set of atoms denoting the set of primitive defect modes (solution states) associated with the leaf nodes of the knowledge sketch.

$\Omega$ is the set of attributes useful in characterizing H and A.

$DCNTX \subseteq \Omega$, is the set of instantiated attributes, generally termed diagnostic contexts. They are extracted from the data records in the problem domain (i.e., AUTORED generated CAD data base).

$\mathring{A}$ is the set of characteristic mappings from H to $\Omega$.

Concerning AVIS' properties, domain attributes, $\Omega$, are generally considered as nominal type with finite and discrete attribute values (Gordon, 1981). They are manipulated on the

basis of set theory. Furthermore, we assume that the attributes associated with the hypotheses are associative and commutative.

Based on these conventions, in the approximate space, $\dot{U}=(D,\dot{A})$, from each observed attribute $m_i$, we can create an equivalence class, denoted $E_{mi}$, of the relation defined by attribute $m_i$. This equivalence class practically includes those hypotheses $h_i$ with attribute $m_i$ as a pattern element of its associated descriptor. Likewise, from diagnostic contexts, DCNTX, containing a set of observed attributes, we can also induce an equivalence class, $E_{DCNTX}$, of the relation defined by the set of attributes contained in DCNTX, where $E_{DCNTX}$ includes the hypotheses $h_i$ having their constituent attributes covered by DCNTX, ie.,

$$E_{mi} = \{ h_i \} \quad \text{with} \quad m_i \epsilon \alpha(h_i)$$
$$E_{DCNTX} = \cup E_{mi} = \{ h_j \} \quad \text{with} \quad m_i \epsilon \text{DCNTX}, \text{ and } m_i \epsilon \alpha(h_j).$$

The relations between attribute and equivalence class can be appreciated from some realistic examples. In medical diagnosis, atom means disease, attribute denotes manifestation, and a hypothesis can represent either a specific disease or a set of suggested diseases. From an observed manifestation $m_i$, the doctor can deduce a set of diseases each including $m_i$ as its disease symptom. This set of diseases, via definition, is the equivalence class of the

relation defined by $m_i$. In AVIS, atom means primitive defect mode, attribute represents symptom feature (diagnostic context), and hypothesis means generic defect category. Based on a symptom feature $\eta_i$, we can induce a set of hypotheses each of which includes the symptom feature $\eta_i$ as an element of its associated descriptor. This set of generic defect modes is the equivalence class of the relation defined by the symptom feature $\eta_i$. The following two tables summarize the above.

Table 3.2 The Concept of Equivalence Class in Medical Diagnosis.

| Domain | Medical Diagnosis |
|---|---|
| Atom | Disease |
| Attribute | Manifestation $m_j$ |
| Hypothesis | single disease or a set of disease |
| Equivalence -Class | induced disease set $\{h_i\}$ with $h_i$ having $m_j$ as its disease symptom |

Table 3.3 The Concept of Equivalence Class in AVIS Application.

| Domain | AVIS |
|---|---|
| Atom | Primitive Defect Mode |
| Attribute | Symptom Feature $\eta_j$ |
| Hypothesis | Generic Defect Category |
| Equivalence -Class | clustered hypotheses $\{h_i\}$ with $h_i$ having $\eta_j$ as its feature pattern |

On the basis of diagnostic contexts, the generated equivalence class could be a huge body of hypotheses. Theoretically, it is a conditional (conditioned on diagnostic contexts) upper approximate of the set of target atoms (the existing atoms, e.g., existing defect modes). Finding this set of hypotheses is not satisfactory simply because, due to the sharing of attributes, some false alarms might be included. As we shall explain below, the major advantage of SICS inference control strategy is the intelligent use of the knowledge sketch to control the creation and elimination of hypotheses in the equivalence class so that the size of the equivalence class could be reduced to the best upper bound (the least upper bound) covering the target atoms, thus achieving a more concise expression. Conventional rule systems fail to incorporate knowledge structure in their inference mechanisms; therefore, the system, whether based on forward chaining or backward chaining, virtually conducts an exhaustive (or opportunistic) search within the huge set of the equivalence class generated by the diagnostic contexts. More observations, according to the rough sets theory, might create a larger equivalence class, and thus more inference branches. Facing so many alternatives, the rule systems reach solutions virtually opportunistically (or by heuristic in conjunction with certain constraints, such as a priori information about the number of disorders, or the assumption that only one refinement is needed).

In terms of the notations of rough sets theory, the inference mechanism of SICS involves the following aspects:

(1) Generate the equivalence class based on the diagnostic contexts.

(2) Generate the best upper bound of the target atoms from the equivalence class.

(3) Find from the reduced best upper bound the best lower bound of the target atoms.

(4) Check whether the best upper bound equals to the best lower bound.

Since the atoms in the knowledge domain presumably are finite and discrete, SICS terminates the inference process when the best upper bound equals the best lower bound. In case it fails to reach this destination (no best lower bound is found), relying on guidance from the knowledge sketch, SICS subsequently acquires new evidences in order to tighten the solution boundary without diverging the reasoning focus in the sense that the newly created best upper bound conditioned on the newly explored evidence tends to be a subset of the formerly identified best upper bound.

## 3.6 Framework of SICS

In the following derivation, we assume that all diagnostic observations, attributes, are nominal, exact, and the characteristic mappings, $\alpha(h_i)$, between hypotheses and attributes are also exact. The issue of dealing with

attributes involving uncertainty will be postponed until the next chapter.

### 3.6.1 Notational Basics

To realize the inference framework just depicted, we utilize the conventions defined below.

Definition 3.1 Hypothesis $h_j$ is said to be a superset of hypothesis $h_i$ if $DSPT(h_i) \subseteq DSPT(h_j)$. The superset is called proper if $DSPT(h_i) \subset DSPT(h_j)$, where $DSPT(h_i)$, $DSPT(h_j)$ denotes the descriptor of hypothesis $h_i$ and $h_j$, respectively. The superset of hypothesis $h_i$ is said to be irredundant if none of its proper subsets is also a superset of hypothesis $h_i$.

For a knowledge sketch organized on the basis of specialization scheme (entities at lower hierarchy levels have more specific meanings), a child node, $h_c$, has more attributes than its parent node, $h_p$; that is $DSPT(h_p) \subseteq DSPT(h_c)$. Thus, by definition, $h_c$ is a superset of $h_p$. The relational order is exactly the reverse of the conceptual subsumption which delineates the knowledge sketch. We have the following proposition that summarizes this situation.

Proposition 3.1 In knowledge sketch, the descendant node is a subset of its ancestor node with respect to conceptual coverage, but is a superset of its ancestor node

in terms of attribute descriptions.

The set, subset relations given in definition 3.1 can be extended and employed to a group of hypotheses as follows:

Definition 3.2 A set of hypotheses, $H_j$, $H_j=\{ h_j \}$ is said to be a superset of the set of hypotheses, $H_i$, $H_i=\{ h_i \}$, if $DSPT(H_i) \subseteq DSPT(H_j)$. The superset is called proper if $DSPT(H_i) \subset DSPT(H_j)$, where $DSPT(H_i)$, $DSPT(H_j)$ denotes the descriptors of hypothesis $H_i$, $H_j$, respectively, and $DSPT(H_i)$ is defined as:

$$DSPT(H_i) = \bigcup_{h_k \in H_i} DSPT(h_k).$$

Definition 3.3 A set of hypotheses H' is said to cover diagnostic contexts DCNTX, $DCNTX \subseteq \Omega$, if DSPT(H') is a superset of DCNTX, ie.,

$$DCNTX \subseteq DSPT(H'),$$

H' is said to be an irredundant cover if none of its proper subsets covers DCNTX.

Definition 3.4 A set of hypotheses, $H^*$, $H^* \subseteq H$, is said a success to the problem of inference $<\mathbb{T},H,A,\Omega,DCNTX,\mathring{A}>$ if:

(1) $H^*$ is an irredundant cover of DCNTX.

(2) $\forall\ h_i,\ h_i \in H^*,\ h_i \in A$.

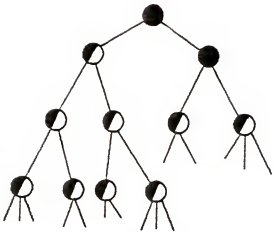
The pragmatism of this definition is obvious. The first condition shrinks the inferred results to the very concise in order to rule out redundancy or false alarms, and the second condition demands that the solution description contain only primitive solution states (e.g., primitive defect modes). SICS, in fact, is the methodology of reaching the conditions specified in this definition, and the whole process starts from the generation of the equivalence class.

### 3.6.2 Generation of Equivalence Class

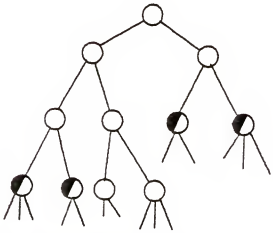
The a priori condition for localizing the best conditional upper bound is the identification of the equivalence class of the relation defined by diagnostic contexts, which theoretically contains all the hypotheses with their associated attributes (fully or partially) covered by the diagnostic. Technically speaking, this equivalence class can be obtained by means of the set of inverse mapping relations $\{\sigma\}$ defined as:

$$\sigma(m_i)=\{c_j\} \quad \text{with} \quad m_i \epsilon DSPT(c_j),$$

where the mapping relation $\sigma(m_i)$ creates the set of hypotheses $\{c_j\}$ with each $c_j$ including attribute $m_i$. $\sigma$ can be derived from $\alpha$ specified in the knowledge domain, such as using the inverted file technique (Wirth, 1976). Conditioned on knowledge sketch $\mathbb{T}$, the created equivalence class can be expressed in Figure 3.3, where symbol ● means all the



Case 1: The Observed Contexts Closely Follow the Attribute Distribution in Knowledge Sketch.



Case 2: Observed Contexts Do Not Follow Well the Attribute Distribution in Knowledge Sketch

Figure 3.3 The Status of Hypothesis Conditioned on Diagnostic Contexts.

associated attributes of the hypothesis have been observed, symbol ◐ means only part of the associated attributes are observed, and symbol ○ means none of the attributes is observed. The status of a hypothesis in the first case is called activated, and activating for the second case, and idle for the third case. They are formally defined as follows:

Definition 3.5 The status of a hypothesis is defined as activated if all of its associated attributes have been observed, or activating, if only part of its associated attributes are observed, or idle if it is neither activated nor activating.

An important relational aspect regarding the propagation of hypothesis status in the knowledge sketch $\pi$ is the following proposition.

Proposition 3.2 In a knowledge sketch organized on the basis of a specialization scheme, the status of the ancestor nodes of an activated node are also activated, while the status of the descendant nodes of an activating node are also activating.

Proof Since the knowledge sketch is organized on the basis of specialization, via proposition 3.1, we have

$$DSPT(h_a) \subseteq DSPT(h_d)$$

where $DSPT(h_a)$, $DSPT(h_d)$ are the descriptor of ancestor node,

$h_a$, and descendant node, $h_d$, respectively. If node $h_d$ is activated, via definition,

$$DSPT(h_d) \subseteq DCNTX$$

therefore,

$$DSPT(h_a) \subseteq DCNTX$$

which proves that node $h_a$ is also activated. Furthermore, if $h_a$ is activating, then

$$DSPT(h_a) \cap DCNTX \neq \phi \quad \text{and} \quad DSPT(h_a) \not\subseteq DCNTX,$$

since $DSPT(h_a) \subseteq DSPT(h_d)$

so $DSPT(h_d) \cap DCNTX \neq \phi \quad \text{and} \quad DSPT(h_d) \not\subseteq DCNTX,$

and we prove the second part of this proposition.

### 3.6.3 Generation of Solution Seed Set

On the course to a success, we have to localize from the equivalence class the best upper bound of target atoms. For this purpose, we use a footstep called solution seed set:

Definition 3.6 A set of hypotheses, denoted $H_{sd}$, is called a set of solution seeds conditioned on knowledge sketch $\pi$ and diagnostic contexts, DCNTX, if

(1) $H_{sd}$ is the maximum hypothesis set constituted by activated hypotheses and activating hypotheses,

(2) None of the activating hypotheses is a superset of another activating hypothesis,

(3) None of the activated hypotheses is a subset of another activated or activating hypothesis.

The solution seed set is termed proper if it meets the following conditions:

Definition 3.7 A set of hypotheses, denoted $H_{psd}$, is called a proper solution seed set if

(1) $H_{psd}$ is a subset of solution seeds,

(2) every hypothesis $h_i$, $h_i \epsilon H_{psd}$, is activated.

Figure 3.4 depicts some realistic aspects about the solution seed set and the proper solution seed set so defined. The worth of using this footstep is the existence of an important relationship between $H_{psd}$ and diagnostic contexts drawn by the following lemma:

Lemma 3.1 Let $H_{psd}$ be a proper solution seed set, and DCNTX be diagnostic contexts, then $DSPT(H_{psd}) \subseteq DCNTX$.

Proof If $h_i$, is activated, then via definition 3.5,

$$DSPT(h_i) \subseteq DCNTX.$$

Since $H_{psd}$ is a proper solution seed set, therefore, every one of its constituent hypotheses is activated.
From the above it follows that

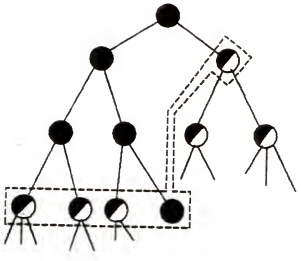
$$\forall h_i,\ h_i \epsilon H_{psd},\ DSPT(h_i) \subseteq DCNTX.$$

Therefore,

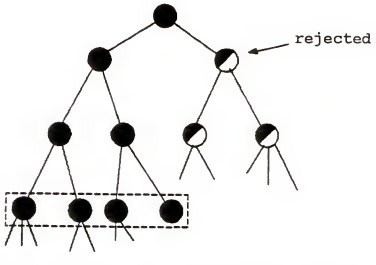
$$DSPT(H_{psd}) = \cup(DSPT(h_i));\qquad h_i \epsilon H_{psd}$$
$$\subseteq DCNTX$$

which proves this lemma.

On the basis of the proper solution seed set, the success



Case 1: A solution seed set contains activated hypotheses and activating hypotheses.




Case 2: A proper set of solution seeds contains only activated hypotheses.

Figure 3.4 Selections of Solution Seeds.

of inference problem in definition 3.4 can be achieved via the following.

Lemma 3.2 A set of hypotheses $H^*$ is a success, if

(1) $H^*$ is a proper solution seed set,

(2) $H^*$ can cover diagnostic contexts DCNTX,

(3) $\forall h_i$, $h_i \epsilon H^*$, $h_i \epsilon$ A.

Proof Since $H^*$ is the set of proper solution seeds, via Lemma 3.1, we have

$$DSPT(H^*) \subseteq DCNTX,$$

on the other hand, if $H^*$ can cover DCNTX, then via definition 3.3, we have

$$DCNTX \subseteq DSPT(H^*).$$

These two relations imply that

$$DSPT(H^*)=DCNTX,$$

which means $H^*$ is an irredundant cover of DCNTX. Since condition (3) specifies

$$\forall h_i, \quad h_i \epsilon H^*, \; h_i \epsilon A,$$

that means all the elements of $H^*$ are atoms. Following the definition of success, $H^*$ qualifies as a success of the solution reasoning.

Lemma 3.2 practically delineates the guidelines for reaching a success of inference. Following this lemma, our inference mechanism is arranged to pursue three objectives: (1) Based on the diagnostic contexts, generate the equivalence

class, $E_{DCNTX}$, and conditioned on knowledge sketch $\mathbb{T}$, create from $E_{DCNTX}$ a solution seed set, $H_{sd}$. These are the major missions of integral analysis.

(2) Expand $H_{sd}$ into a proper solution seed set $H_{psd}$ as the main function of differential diagnosis.

(3) Terminate inference operation when $H_{psd}$ can cover diagnostic contexts and $\forall h_i$, $h_i \epsilon H_{psd}$, $h_i \epsilon A$.

3.6.4 Integral Analysis

As specified in section 3.6.2, the equivalence class of the relation defined by diagnostic contexts can be obtained with the utilization of the inverse mapping relations ($\sigma$). We strategically distribute the elementary sets of the equivalence class along the knowledge sketch (i.e., equivalence class conditioned on knowledge sketch) and those invoked hypotheses, in turn, are sorted into two groups: activated hypotheses and activating hypotheses. Following the second and third condition given in definition 3.6, we delete those activated hypotheses which happen to be the subset of other activated hypotheses, and also those activating hypotheses which are the superset of other activating hypotheses. The remaining hypotheses, thereby, constitute the set of solution seeds.

In case the diagnostic contexts and $H_{sd}$ are all empty (e.g., at the beginning of the inference process), the root node of $\mathbb{T}$ covering the domain universe is selected as the

solution seed. The whole process can be summarized as follows:

1. If the solution seed set $H_{sd}=\{\phi\}$, then set up the initial $H_{sd}=\{h_r\}$, where $h_r$ is the root node of the knowledge sketch and return, else continue.
2. Receive the diagnostic contexts, DCNTX. If DCNTX=$\{\phi\}$, then return, else continue.
3. Based on the inverse mapping relations $\{\sigma\}$, create the equivalence class defined by diagnostic contexts DCNTX, and store that in two lists, List-A=$\{h_i\}$, where $h_i$ is activated hypothesis, List-B=$\{h_j\}$, where $h_j$ is activating hypothesis.
4. Check the knowledge sketch and delete $h_i$ from List-A where $h_i$ has descendants with status activated or activating.
5. Check the knowledge sketch and delete $h_j$ from List-B where $h_j$ has ancestors with status activating.
6. Update the solution seeds $H_{sd}$ containing the remaining hypotheses in List-A and List-B, and return.

### 3.6.5 Differential Diagnosis

The objective of differential diagnosis is to expand the solution seed set, $H_{sd}$, into a proper solution seed set, $H_{psd}$. Since the attribute in the problem domain is assumed to be associative and commutative, false alarms (hypotheses incidentally triggered) might occur due to the sharing of attributes with some target atoms. Those falsely invoked

hypotheses appear in $H_{sd}$ with activating status, and they are examined for confirmation or rejection during the differential diagnosis. The decision is made based on the investigated outcomes of the left-out attributes which are defined as the attributes associated with the activating hypothesis but not covered by the current diagnostic contexts. In AVIS, the left-out attributes are used as commands sent to the context generator in the knowledge base for accessing the anticipated features from the target data record. (In general cases, the left-out attributes might be organized in the form of a questionnaire and sent to the user.) If the feedback answers regarding an investigated hypothesis are confirmatory, the hypothesis changes its status from activating into activated, and is included into the proper solution seed set. Otherwise it is considered as false alarm and ruled out from further consideration. Every activating hypothesis in $H_{sd}$ goes through this process respectively, until only the hypotheses with activated status remain, and thus form the proper solution seed set, $H_{psd}$. The whole process can be summarized as:

1. Set up queue $\delta=\{h_i\}$, $h_i \epsilon H_{sd}$ and $h_i$ has activating status. If $\delta=\{\phi\}$, then return, else continue.
2. Check the knowledge base and determine the left-out attributes associated with $h_i$, where $h_i \epsilon \delta$.
3. Command the context generator in knowledge base to investigate the left-out attributes. If the feedback

attribute value is positive, the hypothesis is activated, otherwise, delete that from $H_{sd}$.

4. Repeat step 2 and step 3 on all the hypotheses in $\delta$.
5. Redefine $H_{sd}$ now containing only activated hypotheses as the proper solution seed set, denoted $H_{psd}$, and return.

### 3.6.6 Termination Check

Following Lemma 3.2, checking for termination consists of two criteria:

(1) The proper solution seed set, $H_{psd}$, obtained in differential diagnosis can cover diagnostic contexts.
(2) Every hypothesis $h_i$, $h_i \epsilon H_{psd}$, is an atom.

If these two conditions are satisfied, the inference operation is concluded, otherwise, the inferred $H_{psd}$ is not conclusive nor reliable and we need to go further (go deeper along the knowledge sketch). To avoid diverging current reasoning focus, strategically only the subnodes of hypotheses already included in $H_{psd}$ are invoked into activating status as the new members of $H_{sd''}$. This policy will guarantee that $H_{sd''}$ be refined and is a proper subset of $H_{psd}$ in terms of the coverage of elementary atoms. It is the way of controlling the creation of a new solution seed set without distorting the promising reasoning direction(s) originally set up. The differential

diagnosis then works on $H_{sd''}$ with the objective of making $H_{sd''}$ into a proper solution seed set $H_{psd}$ as explained in the last two sections. Then the system repeats the operation of termination check. This iteration virtually pumps forward the inference progress with more and more specific hypotheses being included.

One key issue in expanding $H_{psd}$ into $H_{sd''}$ is the selection of which competing hypothesis in $H_{psd}$ should be granted for subnode proliferation. Intuitively, the privilege is awarded to the most promising hypothesis and a prominence index is employed as the measure for this purpose. The prominence index expressing how well a hypothesis, $h_i \epsilon H_{psd}$, can cover current diagnostic contexts DCNTX, is calculated based on the similarity measure between DCNTX and $h_i$ (Tou, 1985). The one which gains the highest score earns the right to proliferate its subnodes and update the solution seed set. The whole process can be summarized as follows:

1. Consult knowledge sketch, if the proper solution seed set, DCNTX $\subseteq H_{psd}$, and $H_{psd} \subseteq A$, then terminate the inference process, else continue.
2. Set up a queue $\delta' = (h_i)$ where $h_i \epsilon H_{psd}$ and $h_i \notin A$.
3. Calculate the prominence index (PI) of each $h_i$ in $\delta'$ and decide the favorable hypothesis $h_j$ with $PI(h_j) \geq PI(h_i)$, $\forall h_i$, $h_i \neq h_j$.
4. Activate and include the child node of $h_j$ into $H_{psd}$, and

redefine $H_{psd}$ as the solution seed set, $H_{sd}$, then return.

### 3.6.7 Functional Organization and Control Stage Transition

Figure 3.5 illustrates the functional flow chart of the whole inference process. For general applications (where the context generator is replaced by the user), a special feature in this reasoning strategy is the capability of processing the pooled observations the user pours into the system at once. As mentioned, associative and commutative are two common features in most realistic problems. Unfortunately, solution search along tree type structures generally does not allow this flexibility, since the searching mechanism generally is top-down and serial in character, i.e., the search starts from the root node, then progressively moves downward. Certain policies for controlling back tracking might be included depending on the details of reasoning strategies (Winston, 1984). But generally speaking, there is an implicit order about attribute investigation. To be more intelligent during man-machine interaction, the system had better accept and process the pooled evidences at a time regardless of the order of attribute appearance. (However, this relaxation in fact offsets the virtues of the knowledge structure.) To deal with this controversial issue, in SICS, the operation contains two stages; passive and active explained as below.

Passive inference control stage. The inference stage is passive when the user inputs a set of pooled evidences

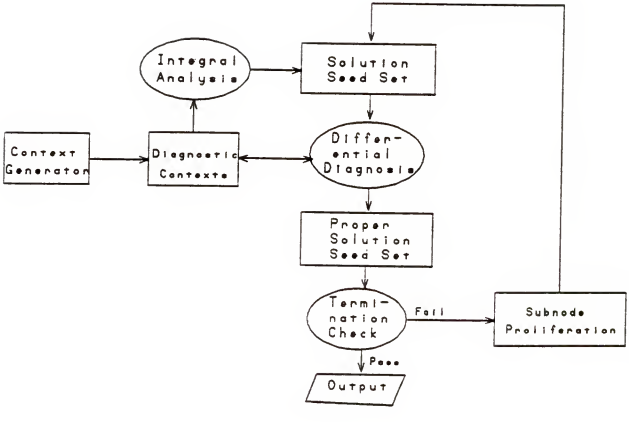



Figure 3.5 Functional Organization of SICS.

which do not coincide with the implicit order specified in the knowledge sketch. Under this situation, the user captures certain initiative and control over the reasoning path, and forces the system to adjust its reasoning steps following the pooled evidences. This stage is important for the system to be friendly and intelligent, especially at the beginning of the inference process when the user might have already acquired certain amount of observations. Most importantly, this flexibility also makes possible the identification of new targets which are not covered in the upper bound computed earlier. Their attributes, newly observed by the user and poured into the system, can trigger a new integral analysis and redefine (broaden) the upper bound of the targets.

Active inference control stage. The inference stage is termed active denoting the situation that the system is actually in charge of the inference process, and the user simply follows the system's instructions, for instance answering prompted questions, fetching data, conducting laboratory tests, etc. This is the traditional scenario, and the system is mainly engaged in differential diagnosis.

Control stage transition. The critical point in the whole inference mechanism design is how the system reacts to the user's pooled observations in the passive reasoning stage, and smoothly transits to the active reasoning stage. This issue is tactically solved in the integral analysis via encoding the pooled observations into activated, and

activating hypotheses, respectively. Those activating hypotheses virtually stock the evidence which are useful either in later stages of the reasoning, or creating new clues regarding the new atoms which have not been identified before. In this way, every piece of evidence can be accepted and processed without hindering the inference concentration or ignoring new target atoms, and there is no need to put restrictions on the timing for the transition to happen. It can be at the beginning of the process, or at any stage of the analysis. The inference mechanism is robust enough to absorb this interruption via switching back and forth between the passive reasoning stage (starting from integral analysis) and the active reasoning stage (alternating differential diagnosis and termination check), hence, compromises between the arbitrariness of attribute appearance in the pooled observations and the rigidity of the semantics in the knowledge sketch. Of course, no efficiency benefit is guaranteed from the frequent interruption of this kind, but the system is ready to perform both purely rigid and adaptive inference.

For the AVIS application, only the active inference control stage is operative since the system is totally in charge of the inference process.

3.6.8 Effectiveness of SICS Inference Strategy

As already emphasized, the induced equivalence class $E_{DCNTX}$ of the relations defined by diagnostic contexts, DCNTX, could be impractically large (e.g., the whole concept space), which virtually defeats the feasibility of exhaustive search for solutions. The following lemma explains the reason why a structured knowledge base is superior to an unstructured one (e.g., rule base). The key contribution a structured knowledge base could make is precisely the directives for generating the best upper approximate of the target atoms from the induced equivalence class.

Lemma 3.3 Conditioned on diagnostic contexts, DCNTX, and knowledge sketch, $\mathbb{T}$, the proper solution seed set, $H_{psd}$, after the differential diagnosis is the best conditional upper approximate of the set of target atoms, X, i.e.,

$$H_{psd} = \overline{Apr}(X)$$

Proof: Classification by virtue of attribute observation is inexact in case that domain attributes are associative and commutative. The realistic sense of the equivalence class defined by diagnostic contexts is the delimitation of an upper approximate of the actual target atoms, which, unfortunately, might be as big as the whole domain space. To simplify solution finding, heuristic or other nonformal means might be necessary. The virtues of knowledge sketch is its strategic partition of the domain universe in

such a way that the coverage of the node (hypothesis) can be hierarchically reduced. Conditioned on knowledge sketch, the members of the equivalence class appear as either activating or activated hypotheses depending on their associated descriptors and current diagnostic contexts. Following the procedure for generating solution seeds (specified definition 3.6), based on proposition 3.1 and proposition 3.2, we reduce the size of plausible solutions into a better upper bound. To further tighten the upper approximate, of course we could arbitrarily or heuristically discard some hypotheses, no matter if activating or activated, from the set of solution seeds. But this arbitrariness runs into the risk of possibly erasing actual target atoms. In practice, we are confident about certain assertions only when the associated hypothesis is activated (confirmed). Therefore, in talking about the best upper bound, only the set of activated hypotheses can make sense. Since the differential diagnosis has mechanism of ruling out all false alarms (falsely invoked hypotheses due to the sharing of attributes with target atoms), that means the size-reduced proper solution seeds are legitimate to specify the scope of coverage, i.e., the best upper bound conditioned on the current diagnostic contexts. Via this argument, the only problem left is the proof of the uniqueness of the best upper bound, i.e., no other set of activated hypotheses other than the proper solution seed set can be the best upper bound.

The proper solution seed set, via definition 3.6, 3.7 and proposition 3.2, contains only those activated hypotheses located at the deepest level in the sketch as shown in Figure 3.4. We claim that this set of hypotheses creates the most restricted scope, thus, the best upper bound of the target atoms. This can be proved by contradiction.

Let $H_{psd}$ be a proper solution seed set, via definition 3.6, 3.7, for hypothesis $h_i$, $h_i \epsilon H_{psd}$, none of its subsets is within this set, also, none of its supersets is activated (otherwise, it will be included). Suppose there is another solution seed set, $H_{psd'}$, specifying the best upper approximate to the target atoms. Following the specialization scheme in organizing the knowledge sketch, $H_{psd'}$ should be sitting at a lower level of $H_{psd}$, which implies $DSPT(H_{psd}) \subseteq DSPT(H_{psd'})$, that means $H_{psd'}$ is a superset of $H_{psd}$, which, however, violates the presumption that $H_{psd}$ is a proper solution seed set. Therefore, we proved that only $H_{psd}$ qualifies the best upper bound of the set of the target atoms.

Another significant feature of SICS is the completeness and conciseness of the solution found. It implies that all existing target atoms can be localized with redundancy. This argument can be summarized into the following lemma:

Lemma 3.4 A success $H^*$ created via SICS is complete and concise under the condition that the attributes associated with the target atoms are observable.

Proof SICS always pursues its proper solution seed set, $H_{psd}$, following available diagnostic contexts. Based on Lemma 3.3, $H_{psd}$ constitutes the best upper bound of the target atoms, that means all the target atoms are always included in $H_{psd}$, thus, the inference result is complete.

Furthermore from the proof of Lemma 3.2, we know a success, $H^*$, once obtained, is an irredundant cover of diagnostic contexts, i.e., $DSPT(H^*)=DCNTX$. Hence the inference result is concise, and we conclude the proof.

The upper bound concept in SICS is a general clustering of solution states, therefore, the inference control is a batch mode of operation. We term this scheme pseudoparallel denoting its difference from the sequential control schemes. Its effectiveness is more significant in dealing with multi-faults diagnosis, where the huge body of hypotheses combination makes the problem potentially computationally exponential. The virtues of SICS is its strategic usage of knowledge sketch in limiting the selection of solution states into a few choices from the huge body of the equivalence class of relations defined by diagnostic context. Since the sought solution is complete and concise, no redundancy or arbitrary guess is included. That means the solution generated by SICS is able to meet the requirement specified by the principle of parsimony (Reiter, 1987) regarding multi-faults diagnosis. The power of SICS, in principle, comes from the knowledge sketch,

a standpoint consistent with the common acknowledgement in the AI community. With the aid of knowledge sketch, SICS is able to pursue its solution finding without diverging its attention. Also a powerful feature of this inference control technique is the capabilities of processing pooled evidence via switching back and forth between the passive and the active inference operations. This mechanism relaxes the constraint of the rigid sequence of attribute observation as requested in Establish/Refined. It also realizes the possibility of exploring new target atoms.

## 3.7 Discussion

The framework of SICS described in this chapter works on observable attributes with certainty. However, it is not uncommon for the case which prohibits these preconditions, for instance the attribute observation is expensive, or prohibitive (e.g., lack of proper instruments), or uncertain. Even time limitation might also trigger the need for rushing to a quick result. Under such a circumstance, complete and precise observation is impossible. SICS, indeed, needs to augment its capabilities to cover this kind of uncertainty calculation. This is exactly the main subject of the next chapter.

# CHAPTER IV
# FORMALISM OF APPROXIMATE REASONING IN AVIS

## 4.1 The Controversy: Bayesian or Non-Bayesian

In Chapter III, we assume all pattern elements, attributes, descriptors, be exact without any ambiguity involved. In dealing with real world problems, this assumption is not always valid. On the contrary, it must be anticipated that the information supplied to the system may be inaccurate or incomplete, because, among other reasons, the sources of the information might be imperfect, or the system may be subject to resources limitation; thus the information is incomplete. Hence, the system faces decisions to be made with uncertain information. There have been many attempts in the past to construct a "calculus of uncertainty" for expert system applications since MYCIN (Shortliffe, 1976) introduced the concept of "Certainty Factor" for belief revision and uncertainty propagation. Belief is concerned with the degree of confidence an agent has in the truth of a particular fact. Since the notion of confidence is very difficult to formalize, most of the measures that have been used are quantitative, for instance by the employment of an uncertainty function. An uncertainty function, P, is usually derived from propositions to a real number such that P(a) indicates the truth of

proposition a. Based on this function, if a is more certain than b, then $P(a) \geq P(b)$. The traditional approach is probability theory which, until recently, has provided the best-developed mathematical framework for dealing with uncertainty. Bayes' rule is the guideline used in this framework for manipulating the uncertainty function:

$$P(H|E_1,E_2,\ldots,E_n) = \frac{P(H)P(E_1,E_2,\ldots,E_n|H)}{P(E_1,E_2,\ldots,E_n)} \qquad (4.1)$$

where $P(H|E_1,E_2,\ldots,E_n)$ is the conditional probability of H given $E_1,E_2,\ldots,E_n$. Two simplifying assumptions are usually taken to resolve the difficulty of determining $P(E_1,E_2,\ldots,E_n)$ and $P(E_1,E_2,\ldots,E_n|H)$ (Peebles, 1980):

(1) The events $E_i$ are assumed to be statistically independent, i.e.,

$$P(E_1,E_2,\ldots,E_n) = P(E_1)P(E_2)\cdots P(E_n), \qquad (4.2)$$

(2) The statistical independence between the $E_i$ continues to hold given H, i.e.,

$$P(E_1,E_2,\ldots,E_n|H) = P(E_1|H)P(E_2|H)\cdots P(E_n|H). \qquad (4.3)$$

Therefore, (4.1) can be simplified into a linear format:

$$P(H|E_1,E_2,\ldots,E_n) = P(H|E_1,E_2,\ldots,E_{n-1})[P(E_n|H)/P(E_n)] \qquad (4.4)$$

Equation (4.4) specifies how the conditional certainty in H, given n pieces of evidence, can be computed from the conditional certainty in H given (n-1) pieces of evidence. An appealing aspect is the correspondence between the propagation of conditional probability and human intuition usually in the fashion of IF (premise) THEN (consequence) (i.e., a knowledge chunk in the form of a production rule). This method of calculating uncertainty in fact is the basic reasoning component in PROSPECTOR, a pioneer expert system for mineral exploration (Duda and Hart and Nillson, 1981).

A prominent character in the formulation of probability is the constraint that

$$P(q) + P(q^c) = 1 \qquad (4.5)$$

which means that whatever uncertainty is missing with respect to a proposition q must be attached to its complement $q^c$. It follows that there is no room in this framework for expressing ignorance in the certainty of a proposition and its complement. Ignorance, unfortunately, is a significant feature in human knowledge and also in the knowledge base of expert systems duplicating humans' expertise. This deficiency virtually motivates the invention of the Certainty Factor in the MYCIN system.

Dempster-Shafer theory (Shafer, 1982, Shafer, 1986) proposes an alternative which replaces statistical dependence concerns with logical dependence and offers a choice other

than the Bayes' rule. The belief function derived from this theory seems able to express ignorance properly, and it allows

$$P(q) + P(q^c) \leq 1.$$

However this method is inherently computationally exponential, which becomes the major obstacle with using this method of pooling evidence from disparate sources (Garvey and Lowrance, 1981). Several methods have been proposed to reduce the computation complexity with restrictive simplifying assumptions but sacrificing accuracy (Garvey and Lowrance, 1981, Barnett, 1981).

Sharply contrasted to the quantitative tradition is Cohen' endorsement theory (Cohen and Grinberg, 1983). He argues that the traditional numerical representation of certainty hides the reasoning that produces them, and thus limits one's reasoning about uncertainty. Moreover, he argues, no clear meaning is given to the number so obtained. His endorsement theory adopts a purely qualitative means for approximate reasoning.

A variety of other methods based on various concerns have also been proposed, such as Pearl' probabilistic network (Kim and Pearl, 1983, Pearl, 1985), Cheeseman's subjective probability (Cheeseman, 1983, Cheeseman, 1985), and Zadeh's fuzzy sets theory (Zadeh, 1978, Kosko, 1986), etc. Generally speaking, each has its own virtues as well as weaknesses, either conceptual or computational, but is effective under the

framework for which it is designed (Kanal and Lemmer, 1986). For instance, the probability theories have long been used and demonstrated excellence in dealing with the propagation of uncertainty in communications, since the randomness of system parameters can be described properly based on certain probabilistic models. However, for applications (e.g., the medical application of MYCIN) where the modeling of uncertainty mechanism is generally complex or difficult to assess, heuristic type Non-Bayesian systems seem to be inevitable. As a matter of fact, some researchers (Chandrasekaran and Tanner, 1986) even strongly express their skepticism about the existence of a unitary uncertainty approach effective in handling problems involving intelligent activities, simply because many informal factors, such as subjective heuristics and unreasonable personal preferences, are deeply involved. As the general Non-Bayesian people, they believe that reasoning about uncertainty is a knowledge-intensive task and domain experts have their own heuristic for dealing with typical kinds of uncertainty (Chandrasekaran and Tanner, 1986). Consequently, approximate reasoning had better be task-specific and justified by the system performance, particularly in the scenario of expert systems.

As is evident from the current literature, the debate about the formalisms of approximate reasoning, Bayesian or Non-Bayesian, is still open-ended. There has been much recent activity and interest in exploring new approaches and there

is still much interest in investigating more traditional avenues (Kanal and Lemmer, 1986). Despite this chaos, a common feature in those proposed techniques is the dedication to the formalism of production rule. Hence, two main themes: combination of evidence, and deductive inference, are usually covered and compared.

For AVIS, we draw our inference logic from pattern recognition principles (this is obvious from the organization of knowledge sketch as well as the SICS inference control strategy). As it is already known, the design concepts for pattern recognition are often motivated by the ways in which pattern classes are characterized and defined. Therefore, they can be mathematical, heuristic, syntactic, or their combination (Tou, 1982). If the behavior of pattern classes can be characterized via probability models, it would be natural to adopt Bayesian approach. The intelligent system, MEDIKS, exemplifies this standpoint. However, in case such prerequisite can not be fulfilled adequately, Non-Bayesian approach can be brought in as an alternative. In AVIS, we encounter the difficulty of formulating a proper probability model, due to the complexity of AUTORED' interpretation mechanism and the involvement of a large number of attributes, hypotheses and messy interrelationships. Hence, we propose a Non-Bayesian method to accomplish the task of approximate reasoning. This uncertainty method, even though like most Non-Bayesian systems, is ad hoc, but the framework is consistent

with AVIS' architecture, especially the bi-polarized knowledge expression as stated in Chapter II. It is derived from the Fuzzy Set theory incorporating the technique of evidence space (Rollinger, 1983). Fuzzy set theory, first proposed by Zadeh, has many properties with sound theoretic groundwork for processing uncertain information. Evidence space, on the other hand, is an intuitive but effective means for reporting contradictory evidence. With the combination, our Non-Bayesian approximate reasoning is computationally feasible and sufficiently consistent. We begin the theoretic basis of this uncertainty approach with the fundamental concepts of evidence space.

## 4.2 Non-Bayesian Approximate Reasoning: Theoretic Basis

### 4.2.1 Evidence Space

In the evidence space, uncertain or contradictory information are expressed via the following form:

$$(EP)(ew_+, ew_-)$$

where EP stands for evidence point regarding the status of uncertainty of a piece of evidence (proposition, attribute, etc.), and $ew_+$, $ew_-$ are the associated positive and negative weighting values with range in [0, 1]. This expression is a summary of two items of information: a report on how strongly the validity of the evidence is believed as shown in a crispy number $ew_+$, and a report on how strongly the validity of the

negation of the same evidence is believed as shown in another crispy number $ew_-$ (Driankov, 1987). A distinctive merit of this scheme comes from its description capability, particularly toward conflicting phenomena. For example, in the evidence space it is very easy to differentiate the following four situations (Rollinger, 1983):

(1) (strong) positive and no negative evidence.

(2) No positive and (strong) negative evidence.

(3) No evidence at all.

(4) (some) positive and (some) negative evidence.

Contrarily, these four situations are rather difficult to distinguish based on the one-dimensional approaches where usually only the difference between the positive and the negative evidential weightings is taken into account. The determination on these two weights, in general, can be mathematical, statistical, or by certain subjective means. They may come from independent sources, but are bounded by the following relationship:

$$(ew_+ + ew_-) \in [0, 2],$$

That is our notation of graded belief in the evidence space does not obey the relationship existing between the probabilistic measures of support. However, this extension relaxes the restriction that our commitment of belief to a particular event implies our commitment of belief to its negation, as is usually valid in the probability framework.

Geometrically, the paired representation is a point located in a two-dimensional space (Rollinger, 1983). Figure 4.1 (a) shows the partition of supportive and adverse zones in this evidence space. Even though the contradiction expression is our main interest, this evidence space, in fact, can also record the distribution of probability and possibility as shown in Figure 4.1(b).

In the AVIS application, on the basis of bi-polarized knowledge expression each defect mode is characterized by two descriptors, one supportive and one adverse (section 2.4.2). The realization of these two descriptors can be considered as the contradictory evidence supporting or rejecting the truth of the defect mode. The same concept is also applicable to attribute instantiation. For instance, during the investigation of the left-out attributes (conducted in the differential diagnosis of SICS as stated in section 3.6.5), if the extracted features confirm the existence of the anticipated attributes, they gain certain supportive weighting, otherwise, they receive adverse weightings. This arrangement accelerates the refutation of the false-alarmed hypotheses. As a matter of fact, the same concept of evidence space can be coherently applied to the whole approximate reasoning framework covering three conceptual levels: attributes, descriptors, and entities.

The description power of this evidential weighting pair, like a fuzzy number, can be further enhanced via incorporating

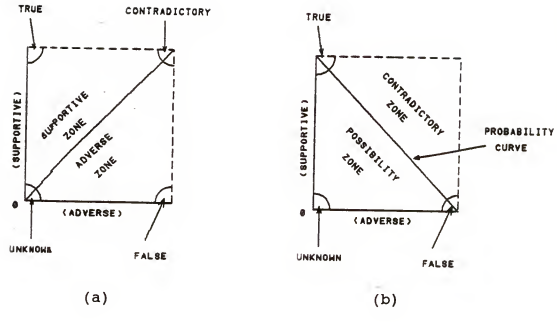



Figure 4.1 Some Aspects of the Evidence Space.

various verbal indicators, such as 'IT IS TRUE', 'IT MAY BE', 'IT IS FALSE', etc. (Rollinger, 1983), but this aspect is not included into the design of AVIS.

### 4.2.2 Pooling Contradictory Evidential Weightings

A problem associated with the paired uncertainty expression is the vagueness in the indication of overall commitment to a decision (e.g., 'go', 'no go'). For instance, suppose we come up with the paired evidential weightings (0.7, 0.2) regarding the defect mode "Denot-Line" (denotation line), it is still not clear how certain we are willing to accept that the defect mode is a denotation line. In other words, this paired uncertainty expression is good at contradiction description, but unlike one-dimensional variable (such as probability), it might not be adequate in decision making, due to the lack of strict order in 2-dimensional space. (Similar difficulty is inherent to other kinds of paired evidential expressions, such as the (belief, plausibility) in Dempster-Shafer theory (Shafer, 1982) or the (necessity, possibility) in the fuzzy set-theoretic measures (Zadeh, 1978). Generally speaking, only a partial order relation can be ranked in this type of expression (Driankov, 1987).) Since AVIS is supposed to make decisions automatically, a crisp number summing up the degree of commitment covering all contradictory evidence seems to be necessary for the system to reach conclusions. We propose that the pooling of contradictory evidence be based

on the Belief Characteristic Function (BCF), a concept drawn from the membership function in fuzzy sets theory (Negota and Ralescu, 1987).

Belief characteristic function: (BCF). A BCF function, $BCF(ew_+, ew_-)$, is a real-valued function from the pairs $(ew_+, ew_-)$:

$$\{ BCF(ew_+, ew_-) \dashrightarrow es \mid ew_+, ew_- \in [-1, 1]; es \in [0, 1] \}$$

where $ew_+$, $ew_-$ are the evidential weightings associated with the evidence, es stands for the evidential strength after pooling $ew_+$ and $ew_-$, and denotes the summarized degree of belief. In AVIS, pooling contradictory evidence is regarded as a matter of background knowledge, personal sensitivity, prejudice, and several other subjective factors. Belief is generally assumed nonadditive (a key feature in subjective information processing (Jumarie, 1986)). Therefore, the functional format of BCF, in general, is a function of our subjectivity. To be in agreement with our intuition, a BCF might be required to satisfy the following conditions:

(1) $BCF(1,0) = 1$; (true),

(2) $BCF(0,1) = -1$; (false),

(3) If $ew_+ = ew_-$, then $BCF(ew_+, ew_-) = 0$; (no bet),

(4) If $a > b$ then $BCF(a, ew_-) > BCF(b, ew_-)$,

(5) If $a > b$ then $BCF(ew_+, a) < BCF(ew_+, b)$.

These conditions imply that the BCF is monotonic and normalized, and the image of BCF is linear scale (on which a strict order can be sought) expressing our confidence from extreme uncertain or disbelief (e.g., -1) to extreme certainty or belief (e.g., +1).

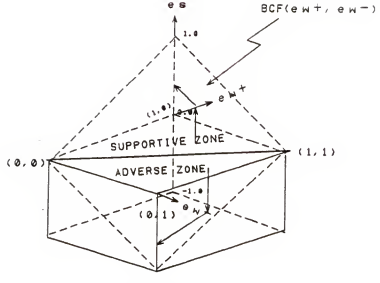
Characterizing our mentality via BCF function is technically similar to the membership function used in fuzzy logic. We might even consider the BCF function as a two-dimensional membership function. Moreover, BCF function also has the flavor of a signature table (Samuel, 1967), for converting granularity in BCF can be carefully adjusted and tabulated to match our mental favoritism. Figure 4.2 illustrates two qualified BCF functions; one is <u>even favoritism</u> with the following BCF function

$$es = BCF(ew_+-ew_-) = 1- (1-ew_+) -ew_-. \qquad (4.10)$$

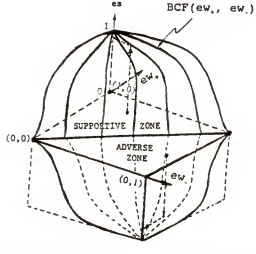
The other is <u>favoring majority</u> with a nonlinear BCF function

$$es = BCF(ew_+-ew_-) = (1-ew_-)^{\alpha} - (1-ew_+)^{\alpha}, \quad \alpha \geq 0. \qquad (4.11)$$

where $\alpha$ is used to adjust the curve shape of the BCF. In general, the image of BCF is not a linear function of the distance between $ew_+$ and $ew_-$. As evident from the BCF shown in Figure 4.2(b), the location of $(ew_+, ew_-)$ is the factor determining the ultimate belief score. That is the BCF




(a) BCF function with even favoritism.




(b) BCF function favoring majority.

Figure 4.2 Two BCF Functions for Pooling Contradictory Evidential Weightings into An Evidential Strength.

function still retains the capability of distinguishing various situations of uncertainty. In AVIS, (4.10) is chosen as the system's BCF function.

### 4.2.3 Uncertain Pattern Matching

Two reasons motivate the incorporation of evidence space and BCF function into our Non-Bayesian framework: (1) to provide an adequate means for recording uncertain and contradictory information, (2) to embed the humans' subjectivity via BCF function into the mechanism of decision making which, as believed by the Non-Bayesian people, should be knowledge-intensive and personalized. Another foundation of our approximate reasoning framework is evidence combination via the technique of uncertain pattern matching. Pattern matching in SICS is to generate the control parameter, prominence index (section 3.6.6), for various usages including solution seed set proliferation. Conceptually, we can view the pattern matching between two pattern vectors, U and V, containing nominal and exact pattern elements as a function of matching degree and mis-matching degree (Hayes-Roth, 1978). We denote the result as PM(U,V)=(a,b,c),

where:

PM: pattern matching operator.

a: the number of pattern elements which are shared by U and V.

b: the number of pattern elements existing in U but not in V.

c: the number of pattern elements existing in V but not in U.

The matching and mis-matching degree, also termed similarity, and dissimilarity, respectively, are calculated as (Gordon, 1981)

$$\text{Degree of Similarity} \quad S = \frac{a}{a+b+c} \tag{4.12}$$

$$\text{Degree of Dissimilarity} \quad D = \frac{b+c}{a+b+c} \tag{4.13}$$

Since pattern elements are assumed exact, from (4.12), (4.13), we have D = 1 - S. In case the pattern elements are uncertain, we might modify the 3-tuple vector (a,b,c) via the following reasoning line.

Suppose a pattern element "g" is shared by U and V with the corresponding evidential strength (i.e., the pooled degree of belief after the BCF function) $es_u$ and $es_v$, respectively. Then the implication that pattern element g is simultaneously existing in U and V is the same as the logic-AND on the proposition that "(g is in U) and (g is in V)." It can be defined based on the logic-AND operation under the MIN Model in fuzzy set theory (Zadeh, 1973) as:

$$STR(g(es_u),g(es_v)) \begin{cases} = \min(g(es_u),g(es_v)), & \text{with } es_u es_v > 0. \\ = 0, & \text{with } es_u es_v \leq 0. \end{cases} \tag{4.14}$$

where STR denotes the implication strength after the logic operation. (When $es_u es_v \leq 0$, the evidences are against each other, so the strength value is set to 0.) Following the same token, when all pattern elements of U and V are uncertain, the 3-tuple vector is calculated by:

$$a = \sum_{g_i \epsilon U,V} |\min(g_i(es_u),\ g_i(es_v))|$$

$$\text{with} \quad (g_i(es_u) \epsilon U) \cap (g_i(es_v) \epsilon V) \cap (es_u es_v > 0) \qquad (4.15)$$

$$b = \sum_{g_i \epsilon U} g_i(es_u)$$

$$\text{with} \quad [(g_i(es_u) \epsilon U) \cap (g_i \notin V)] \quad \text{or}$$
$$[(g_i(es_u) \epsilon U) \cap (g_i(es_v) \epsilon V) \cap (es_u es_v < 0)] \qquad (4.16)$$

$$c = \sum_{g_i \epsilon V} g_i(es_v)$$

$$\text{with} \quad [(g_i \notin U) \cap (g_i(es_v) \epsilon V)] \quad \text{or}$$
$$[(g_i(es_u) \epsilon U) \cap (g_i(es_v) \epsilon V) \cap (es_u es_v < 0)] \qquad (4.17)$$

The similarity and dissimilarity between U and V can be computed following equations (4.12) and (4.13).

Pattern matching, involving a set of pattern elements, in fact, is the way of evidence combination in our uncertainty method. Its main purpose is to summarize all evidential factors provided by the diagnostic contexts, and the result shown as similarity score is used as a control parameter for decision making. The evidence space, BCF function, and uncertain pattern matching, three uncertainty aspects,

virtually constitute the theoretic basis of our approximate reasoning framework.

## 4.3 Uncertainty Calculation Cycle

A careful examination reveals that three distinctive reasoning phases involving uncertainty actually exist in the SICS inference control strategy. They are (1) assessing diagnostic contexts (extracting symptom features from the target data records), (2) pattern matching the descriptors of entities (hypotheses) with the diagnostic contexts, and (3) confirming hypotheses based on their paired descriptors (under the scheme of bi-polarized knowledge expression). Therefore, the approximate reasoning in AVIS is decomposed into three different levels: (1) quality measure of attribute, deciding the reliability of the extracted attribute, (2) realization quality of descriptor, deciding how well the descriptor of hypothesis matches with the diagnostic context (accumulated evidence), and (3) confirmation degree of hypothesis, representing how well a hypothesis has been confirmed based on the pair of realized descriptors. The confirmation degree calculated finally is the prominence index used in the differential diagnosis of SICS for comparison among competing hypotheses. These three reasoning aspects are defined as an uncertainty calculation cycle, and they can be carried out based on the three foundation concepts: evidence space, BCF

function, and uncertain pattern matching, in a coherent manner as interpreted in the following.

### 4.3.1 Attribute Quality Measure

The quality of attribute is defined as the certainty of the extracted attribute. It is measured each time the context generator of the system extracts out a piece of symptom feature (attribute) from the problem domain. If the uncertainty regarding the observed evidence is expressed by a pair of evidential weighting factors ($ew_+$, $ew_-$), then its quality can be measured based on the technique of the BCF function as shown in Figure 4.3. In the current version of AVIS, $ew_+$, $ew_-$, are estimated by a simple mechanism with binary logic. The physical value of an attribute variable is quantized into nominal value based on the thresholds used in AUTORED (e.g., the line length in terms of pixel number is converted into attribute values; long, short, with threshold set to 7 pixels). (An improved approach might be to calculate $ew_+$, $ew_-$ based on the fuzzy number concept.) The BCF function with even favoritism as shown in (4.10) is employed. The set of attributes after this process is the diagnostic context involving uncertainty information.

### 4.3.2 Descriptor Realization Quality

The realization quality of descriptor is defined as the matching degree between the descriptor associated with the

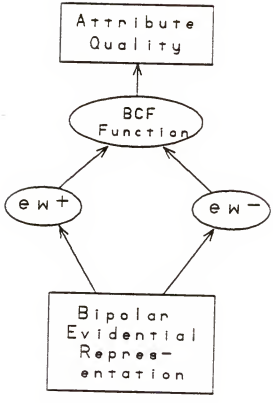



Figure 4.3 Functional Flow Chart of Attribute Quality Measure.

hypothesis and the diagnostic contexts. This measure is carried out based on the technique of uncertain pattern matching including equations (4.15-4.17). To coincide with the expression of AVIS' knowledge sketch, these equations are further modified to include the following aspects: (1) the diagnostic weighting factors associated with the descriptor are included into the computation of the 3-tuple vector, and (2) the evidential strength of each pattern element of descriptor is assumed 1.0. As stated in section 2.4.2, the diagnostic weighting factor (range from 0 to 1) represents the material significance an attribute has in the assertion about the truth of the associated entity (hypothesis). It is the only means used in AVIS to record knowledge uncertainty. In this scheme we implicitly assume that all pattern elements in the descriptor are exact. The way of expressing knowledge uncertainty only through the feature profile of the descriptor is inconsistent with our intuition and also makes the knowledge imprecision tractable. Taking the diagnostic weighting factors into account, the 3-tuple vector given in section 4.3.3 is modified into the following form:

$$a = \sum_{g_i \in DSPT(h),\, DCNTX} \left| \min\left(g_i(es_{DSPT})\, w_{DSPT}(g_i),\ g_i(es_{DCNTX})\, w_{DSPT}(g_i)\right) \right|$$

$$\text{with} \quad (g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i(es_{DCNTX}) \in DCNTX) \cap (g_i(es_{DSPT})\, g_i(es_{DCNTX}) > 0) \qquad (4.18)$$

$$b = \sum_{g_i \in DSPT(h)} g_i(es_{DSPT})\, w_{DSPT}(g_i)$$

with $[(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i \notin DCNTX)]$ or
$[(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i(es_{DCNTX}) \in DCNTX) \cap (g_i(es_{DSPT})\, g_i(es_{DCNTX}) < 0)]$ (4.19)

$$c = \sum_{g_i \in DCNTX} g_i(es_{DCNTX})$$

with $[(g_i \notin DSPT(h))) \cap (g_i(es_{DCNTX}) \in DCNTX)]$ or
$[(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i(es_{DCNTX}) \in DCNTX) \cap (g_i(es_{DSPT})\, g_i(es_{DCNTX}) < 0)]$ (4.20)

where $w_{DSPT}(g_i)$ is the diagnostic weighting factor associated with pattern element (attribute) $g_i$ of the descriptor (DSPT(h)), DCNTX denotes the diagnostic context containing the set of instantiated attributes, and $g_i(es_{DSPT})$, $g_i(es_{DCNTX})$ denote the associated evidential strength of $g_i$ in DSPT(h) and DCNTX, respectively.

Taking into account the second factor (all the attribute evidential strength in the descriptor are set to 1.0), equations (4.18), (4.19), (4.20) can be further simplified into:

$$a = \sum_{g_i \in DSPT(h), DCNTX} g_i(es_{DCNTX})\, w_{DSPT}(g_i)$$

with $(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i(es_{DCNTX}) \in DCNTX) \cap (g_i(es_{DSPT})\, g_i(es_{DCNTX}) > 0)$ (4.21)

$$b = \sum_{g_i \in DSPT(h)} w_{DSPT}(g_i)$$

with $[(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i \notin DCNTX)]$ or
$[(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i(es_{DCNTX}) \in DCNTX) \cap (g_i(es_{DSPT})\, g_i(es_{DCNTX}) < 0)]$ (4.22)

$$c = \sum_{g_i \in DCNTX} g_i(es_{DCNTX})$$

with $[(g_i \notin DSPT(h))) \cap (g_i(es_{DCNTX}) \in DCNTX)]$ or
$[(g_i(es_{DSPT}) \in DSPT(h)) \cap (g_i(es_{DCNTX}) \in DCNTX) \cap (g_i(es_{DSPT}) g_i(es_{DCNTX}) < 0)]$ (4.23)

The similarity measures are also modified:

Degree of Similarity
$$S = \frac{a}{\sum_{g_i \in DSPT(h)} w_{DSPT}(g_i) + c} \quad (4.24)$$

Degree of Dissimilarity
$$D = \frac{b+c}{\sum_{g_i \in DSPT(h)} w_{DSPT}(g_i) + c} \quad (4.25)$$

The calculated S and D, in turn, are considered as two contradictory evidential weightings regarding the truth of the "realization of descriptor." Hence, the technique of BCF mapping is applied again to sum up our realistic mentality toward this contradiction. Here, a different BCF function is chosen to convert the (S,D) pair into a quality measure ranged from 0 to 1 (1:= perfect, 0:= no commitment), representing the quality of realization of the intended descriptor. Figure 4.4 depicts this functional aspect, and equation (4.27) is the BCF function employed in AVIS for this purpose.

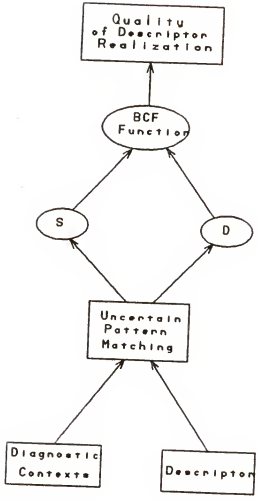



Figure 4.4 Functional Flow Chart of Descriptor Realization Quality.

$$g_i(es) \begin{cases} = 1-(1-g_i(ew_*))-g_i(ew_-); & \text{with } g_i(ew_*) > g_i(ew_-) \\ = 0 & \text{otherwise} \end{cases}$$

(4.27)

4.3.3 Hypothesis Confirmation Degree

As illustrated in section 2.4.2, the knowledge expression in AVIS is fine-tuned (via the scheme of bi-polarized knowledge expression) such that each entity is characterized by two descriptors, one supportive, one adverse. The hypothesis confirmation degree is defined as the evidential strength summarizing the realization qualities of two contradictory descriptors associated with the intended hypothesis. The value is within [-1, 1]. Since a hypothesis is endorsed by two contradictory descriptors, the concepts of evidence space and BCF function can be applied to the calculation of the certainty of the hypotheses. Figure 4.5 illustrates this functional aspect.

4.3.4 Uniformity of Uncertainty Calculation Cycle

A special feature throughout the whole process in the Non-Bayesian framework just presented is "uniformity" in the sense that the aspects of evidence space as well as the concept of BCF for recording and pooling contradictory evidence can be alternatively applied at different levels of knowledge abstract as shown in Figure 4.6. This aspect also

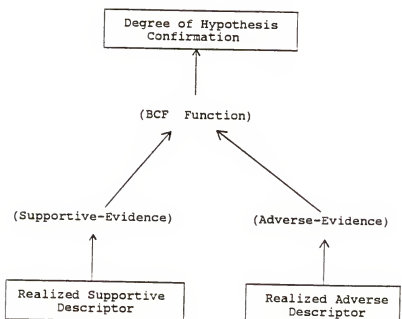



Figure 4.5 Functional Flow Chart of Hypothesis Confirmation Degree.

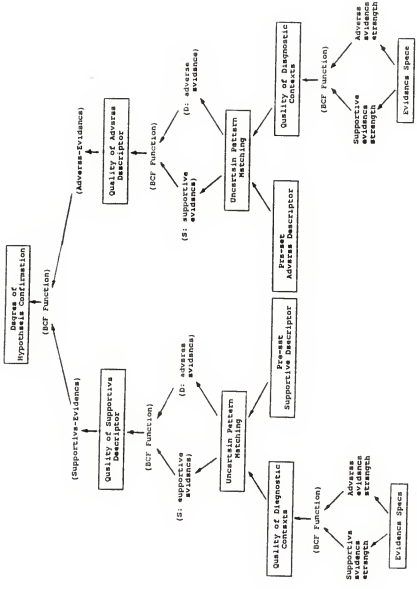



Figure 4.6 Uniformity in Uncertainty Calculation Cycle

demonstrates the conceptual coherence in the Non-bayesian framework. As a matter of fact, there is no mere coincidence for this to happen. As far as uncertainty calculus is concerned, attribute instantiation, descriptor realization, and hypothesis confirmation are conceptually equivalent. In general we can keep three conceptual levels distinct via choosing different BCF functions at different stages.

Example: Suppose a competing defect mode $d_i$ is characterized by two descriptors:

Supportive-DSPT = $<g_1(0.9), g_2(0.6), g_3(0.8)>$

Adverse-DSPT = $<g_4(0.2), g_5(0.3), g_6(0.9)>$

where $g_1$ ,.., $g_6$ are the symbolized attributes, and the figures inside the parentheses (e.g., (0.9)) are the associated diagnostic weighting factors.

Suppose the system has assessed the diagnostic contexts containing the following attributes with their paired evidential weightings included, e.g., $g_1'(ew_+,ew_-)$, as:

$g_1'(0.9,0.2)$
$g_2'(1.0,0.0)$
$g_3'(1.0,0.0)$
$g_5'(0.5,0.0)$
$g_9'(0.3,0.0)$

where $g_9$ is an unrelated evidence. Following the uncertainty calculation cycle just mentioned, we first calculate

the instantiation quality of each attribute. Suppose we choose the BCF function with equal favoritism as shown in equation (4.10). With the notations changed we have

$$es_{gi} = 1-(1-(ew_{+})_{gi})-(ew_{-})_{gi};$$
$$\text{with } 0 < ew_{+},\ ew_{-} < 1 \qquad (4.26)$$

where $es_{gi}$ denotes the pooled evidential strength of attribute $g_i$. Following (4.26), the instantiation quality of each attribute can be calculated as:

$es_{g1} = 1-(1-0.9)-0.2 = 0.7$
$es_{g2} = 1-(1-1.0)-0.0 = 1.0$
$es_{g3} = 1-(1-1.0)-0.0 = 1.0$
$es_{g4} = 1-(1-0.5)-0.0 = 0.5$
$es_{g9} = 1-(1-0.3)-0.0 = 0.3$

The second stage in the uncertainty calculation cycle is the measure of realization quality of descriptors. Via (4.21), (4.22), and (4.23), we first calculate the 3-tuple vector characterizing the matching situation between the supportive descriptor and the observed diagnostic contexts as

$a = (0.7)(0.9) + (1.0)(0.6) + (1.0)(0.8) = 2.03$
$b = 0.0$
$c = 0.5 + 0.3 = 0.8$

Therefore, via equation (4.24), (4.25), the similarity and the dissimilarity scores are:

$$S = \frac{2.03}{(0.9+0.6+0.8)+0.3} = \frac{2.03}{2.6} = 0.7808$$

$$D = \frac{0.8}{2.6} = 0.3077$$

These two values are the paired evidential weightings regarding the supportive descriptor. In order to obtain the realization quality we make use of another BCF function shown in equation (4.27) to pool together (S, D). Likewise, after changing notations, we have:

$$es_{s\text{-}DSPT}\begin{cases} = 1 - (1-S) - D; & \text{with } S > D \\ = 0 & \text{otherwise} \end{cases} \qquad (4.27)$$

Therefore, the realization quality of the supportive descriptor, $es_{s\text{-}DSPT}$, is:

$$1-(1-0.7808)-(0.3077)=0.4731$$

Similar procedures are applied to the case of the adverse descriptor. Via equation (4.21), (4.22), (4.23), we get

$$a = (0.5)(0.3) = 0.15$$

$$b = 0.2 + 0.9 = 1.1$$

$$c = 0.5$$

Then following equation (4.24), (4.25), we obtain S = 0.079 and D = 0.7894. According to equation (4.27), the realization

quality of the adverse descriptor is 0. Thus, the evidential weighting pair associated with the defect mode is $d_i(0.4731,0)$.

The last step in the uncertainty calculation cycle is to compute the degree of hypothesis (defect mode) confirmation. Suppose we choose again the BCF function shown the equation (4.26) to pool together the high-level evidential weightings, then we get the degree of confirmation about the intended defect mode as:

$$1-(1-0.4731)-0.0 = 0.4731$$

In practical inference we may have several competing hypotheses at a time; the degree of confirmation obtained is the prominence index for comparison.

4.3.5 Dilution Effect

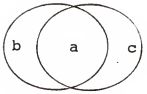
A dilution effect might happen when we apply this approximate reasoning technique to the general multi-faults diagnosis. This effect is caused by the simultaneous appearance of the attributes belonging to other competing hypotheses, and is more significant when the number of competing (and existing) hypotheses is large. Those attributes being relevant to other hypotheses but the current one, tend to dominate the denominator of equation (4.24) and (4.25) via creating a large value of "c" following equation (4.23). This

decreases the score of S as well as the confirmation degree of the intended hypothesis. This dilution effect is symbolically depicted in Figure 4.7. The example in the previous section illustrated this situation also (i.e., the dilution effect caused by $g_9$). A suggested treatment is to use the normalized "a" (the value obtained from equation (4.21) but normalized by $(\Sigma\, w_a(g_i);\ g_i \epsilon DSPT(h))$ as the measure of the realization quality of the descriptor. This compensation effect is also shown in Figure 4.7. Obviously the similarity score might be exaggerated due to the negligence of other non-supportive evidence. The side-effect, however, is reduced when the adverse descriptor also receives the same treatment. We then measure the degree of hypothesis confirmation by pooling together the quality measure of the supportive descriptor and that of the adverse descriptor directly. Following this approach, the quality of supportive pattern vector in the last example becomes

$$a = \frac{(0.7)(0.9)+(1.0)(0.6)+(1.0)(0.8)}{(0.9+0.6+0.8)}$$
$$= 0.8826$$

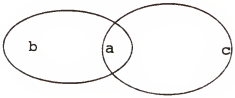
and the realization quality of the adverse descriptor is:

$$a' = \frac{(0.5)(0.3)}{(0.2+0.3+0.9)}$$
$$= 0.107$$




$$S = \frac{a}{a+b+c}$$

(a) Dilution Effect Is Not Significant When Irrelevant Attribute Are Few.




$$S = \frac{a}{a+b+c} \qquad S' = \frac{a}{a+b}$$

(b) Dilution Effect Is Significant When The Number of Attributes Belonging To Other Categories Are Large, and S' Is Better Than S for Similarity Measure.

Figure 4.7 Dilution Effect In Multi-Faults Diagnosis Problems

Thus, the evidential weighting of the defect mode is

$$h_i(0.8826, 0.107)$$

and the degree-of-confirmation is calculated as:

$$1-(1-0.8826)-0.107=0.7756$$

## 4.4 SICS Inference Strategy Augmented

Since the scheme of uncertainty calculus mentioned above is specialized to SICS inference logic, the details of the SICS inference strategy depicted in the last chapter only need a minor modification. Basically the whole functional structure remains effective. The procedure employed in the differential analysis for creating the proper solution seed set is still the same, but the policy granting hypothesis $h_i$ for changing status from activating into activated is decided by the measure of "degree of hypothesis confirmation." If the value is above a threshold (e.g., 0), the hypothesis $h_i$ survives, otherwise, it is unfavorable and is discarded. Another modification is in the procedure for termination check. If the investigation about the left-out attributes are affirmative, the investigated attributes, of course, gain positive evidential weight (e.g., $ew_+=1.0$, $ew_-=0.0$), otherwise, negative evidential weight (e.g., $ew_+=0.0$, $ew_-=1.0$). These weighting values are pooled together via the mechanism of the BCF

function, and the resultant evidential strengths are used for various quality calculations in the uncertainty reasoning cycle. Since the dilution effect in AVIS is not significant, we actually use equations, (4.21) to (4.25), to calculate the degree of hypothesis confirmation.

## 4.5 Summary

Approximate reasoning is an indispensable but also controversial aspect of knowledge systems. The debate about the propriety of formalism, Bayesian or Non-Bayesian, has been forceful, spirited (Cheeseman, 1985, Shafer, 1986) but indecisive. Regarding the AVIS application, we also propose an Non-Bayesian method for approximate reasoning. It is constructed on three uncertainty manipulation concepts: (1) Evidence Space- using a paired uncertainty expression facilitating an intuitive and flexible way for recording uncertain information, (2) Belief Characteristic Function (BCF)- an operator mirroring the human's mental favoritism for defuzzing the ambiguity recorded in the evidence space, and (3) the technique of uncertain pattern matching, based on the MIN model of fuzzy logic, for evidence combination. In SICS' inference logic, the fundamental mechanisms of approximate reasoning is defined as an uncertainty calculation cycle consisting of three steps: attribute quality measure, pattern vector quality measure, and hypothesis confirmation degree measure. The whole calculation cycle can be carried out in a

coherent way in the sense that the concept of the evidence space and BCF function are employed at three different conceptual levels in a uniform fashion.

Another advantage of the Non-Bayesian framework is the incorporation of the knowledge structure (e.g., knowledge sketch) in approximate reasoning. Since humans' subjective tendency granting the decision of an uncertain reasoning is difficult to express purely numerically, the usage of the knowledge structure is nothing more than providing a "compiled" decision policy reflecting the expert's strategies toward problem solving as well as his betting character. A merit of this arrangement is that only a limited number of hypothesis-evidence relationships, under the directives of the knowledge structure, need to be taken into account at each decision step. In other words, the knowledge structure virtually creates a judicious combination of categorical and approximate reasoning- the former is to establish a sufficiently narrow context, and the latter is to make comparisons among hypotheses (Szolovits and Pauker, 1978). In contrast to this standpoint, the techniques for uncertain reasoning adopted in the rule-based systems (with unstructured knowledge base), no matter Bayesian or Non-Bayesian, are inclined to operate in an unsorted evidence universe; therefore, in theory, all kinds of interactions need to be explored and processed. This might be feasible but only when the domain is closed and small.

# CHAPTER V
# SYSTEM INTEGRATION AND PERFORMANCE

## 5.1 System Integration

### 5.1.1 Expert Module, Observer Module, and Control Module

The AVIS system is designed to perform three tasks: data completeness check, data consistency check, and interconnection verification. As mentioned in Chapter I, the configuration of AVIS contains three major parts: a structured knowledge base, an inference mechanism, and a validation controller. The system organization is depicted in Figure 5.1. The heart of the AVIS system is the structured knowledge base which is constituted following a sketch-details scheme. Knowledge sketch is a relational hierarchy representing the strategic outlines of the domain knowledge. Knowledge details consisting of all files of the knowledge base is the supporting unit of knowledge sketch. Also included in this knowledge base is the context generator facilitating AVIS' self-sensing capability for diagnostic context generation. This architecture is able to marry symbolic reasoning with numerical computing into the system. The detailed design has been investigated in Chapter II. Technically, we call the knowledge sketch expert module characterizing its special

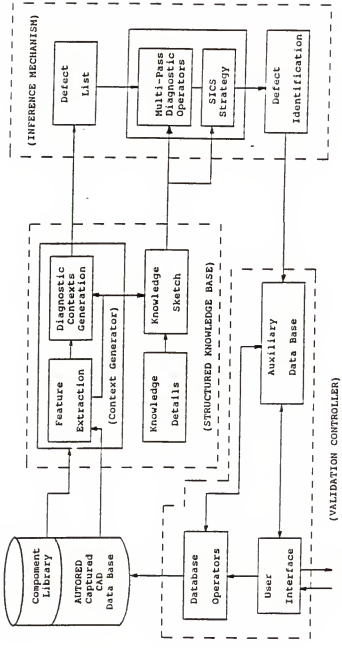



Figure 5.1 System Organization of AVIS

expertise for defect inference, and the context generator observer module denoting its capabilities of symptom feature extraction. In the current version of AVIS, 30 defect modes (7 from the line domains, 15 from the junction domain, and 8 from the component domain) are documented in the expert module. The system also has four groups of context generation programs (line group, junction group, component group, and topology group) containing a total of 17 procedures; in addition 44 symptom features are included in the observer module.

The second system component is the inference mechanism which is the control module of AVIS. Its functional objectives include defect localization and identification. The SICS inference strategy in conjunction with the formalism of approximate reasoning elaborated in Chapter III and Chapter IV is employed for this purpose. This module decides the inference direction and commands the observer module to elicit relevant symptom features from the background data. The control strategy, SICS, is an adaptive beam search with data-driven beam-width adjustment. In operation, the control module, based on the SICS inference control strategy, consults with the expert module for integral analysis. Since no user-interaction is allowed during inference, it conducts the inference process starting from the root node of the knowledge sketch. Based on the feature profile in the associated descriptor, the control module commands the observer module

to investigate the left-out attributes (defined in section 3.2). Depending on the acquired diagnostic contexts, it manages to create a solution seed set containing all plausible solution states. The differential analysis of SICS, in turn, generates the proper solution seed set following the procedure stated in section 3.6.5. Then the termination check, as elaborated in section 3.6.6, decides to conclude the process or to proceed to subnode proliferation. In the latter case, it creates a new solution seed set by including into the original solution seed set those plausible and prominent subnodes specified in the knowledge sketch. Therefore, the system is able to move to a deeper level of the knowledge hierarchy and the differential analysis iterates its process, until a specific defect mode is identified by the termination check. Once the control module successfully identifies the error mode, it then instructs the validation controller to take adequate actions to remedy the error. The organization and function of the validation controller are explained as follows.

### 5.1.2 Validation Controller

Validation controller is the module taking actions to compensate the identified discrepancies in the data base. Three units are included in this module: (1) Auxiliary Database, (2) User Interface, (3) Database Operators.

Auxiliary database. The auxiliary database is the working memory facilitating a "buffer zone" between the AUTORED-generated database and the knowledge base. Any data modification will be temporarily held in this area. It is arranged to prevent the nonmonotonicity feature in the problem domain from altering the contents in the original database.

User interface. This unit is the communication channel between AVIS and the circuit designer. Two facilities; one for explanation, one for arbitration, are included:

(a) explanation facility--after verification, Avis summarizes its results and prompts messages to the user including:
    - the verification statistic,
    - a list of suggested therapies for identified errors,
    - the schematic of the suggested therapies for visualization.

(b) arbitration facility--following the display of the results, the user can have several options including:
    - fully accept the verified results,
    - partially accept and partially override AVIS' decisions, including on-sight modification on the original design,
    - abort the verification process.

These three options provide the user enough flexibility to judge the verification results or further update his/her original design.

Database Operators. This facility is designed to modify the contents of the CAD database; thus it completes the verification operation. The database operators are represented by the format (Tou et al., 1987):

$$OP_i(p_1, p_2, \ldots, p_n)$$

where $OP_i$ denotes the database operator and $p_i$ denotes the $i^{th}$ parameter of the database. Some of the suggested database operators are:

| | |
|---|---|
| REPLDB((a,b),(a,c)):= | replace the original data in data record with new value. (a is file No., b is data No.) |
| DELETE(A,b):= | delete data record b in file a. |
| ENTER(a,b,v):= | enter new data record b (with data contents v) in file a. |

The facilities in this unit are designed to be friendly to the user. An interactive programming environment in conjunction with certain computer graphics utilities for ease of visualization will realize this objective.

### 5.1.3 Multi-pass Diagnosis

A troublesome issue in AVIS, as we have explained in Chapter I, is the nonmonotonicity property embedded in the intended problem domain. Nonmonotonicity is a feature which offsets the conservation of belief. New information might cause an old conclusion to be withdrawn or defeated. Take the

classical example:

```
formal belief:  bird can fly
fact:   penguin is a bird
deduction: penguin can fly
new information: we observe that a penguin can not fly
belief modification: not every bird can fly
```

In AVIS, we encounter similar situations because some circuitry symbols have features of a global type but are constructed from local type properties. Therefore, some unexpected symptom interference might be created due to the obscurance or modification of other local features. For instance, the validation procedure for the superfluous line is the following:

```
THERAPY-LCP5:
        IF (Line# is Superfluous-Line)
        Then (Remove Line#),
```

However the removal of this line might cause a "normal" junction, which is normal due to the existence of that superfluous line, change status into "superfluous." Figure 5.2 illustrates this situation. This phenomenon obviously complicates the problem. Nonmonotonicity has been discussed at length in the AI community. However, instead of following the well publicized techniques, such as Truth Maintenance System (Doyle, 1979) whose back-tracking procedures are quite involved, we tackle this issue based on a multi-pass diagnosis policy. We notice that AUTORED's schematic interpretations are

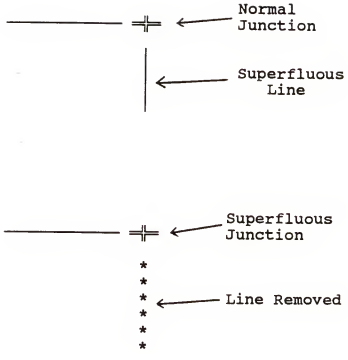



Figure 5.2 The Status of A "Normal" Junction Is Changed after Removing the Superfluous Line.

mainly made on the basis of line signatures, thus, it would be more productive to diagnose line symptoms before others. This diagnostic priority is just one aspect of the multiple-pass diagnosis policy. Three heuristic are included, such as (1) preferential verification, (2) data buffering of intermediate results, (3) human partnership.

Preferential verification. This heuristic specifies the priority of verification as line file ahead of junction file, and junction file before component file. It is arranged with the objective of minimizing the interference from the low level features to the high level ones, thus, reducing the necessity of truth maintenance and also the possibility of back-tracking.

Data buffering. This heuristic requires the verified results be held in the working memory (the auxiliary database inside the validation controller). It is the necessary step to prevent the nonmonotonicity property from destroying the original data base.

Human partnership. The third heuristic formalizes the partnership between human and machine. As a common acknowledgement, for expert systems to function properly, at least two prerequisites must be enforced: (1) domain specific and (2) user corporation. An expert system can not answer arbitrary questions except those it is designed for. A medical consultation system designated to kidney diseases can never

perform well in diagnosing heart diseases. That is, the expert system is always domain-oriented. The second condition implies that no expert systems can deliver a reasonable verdict on the basis of a false confession. As a matter of fact, it is very easy to confuse and make the system fail "gracefully" if the user deliberately pours into the system false or inconsistent data. These two requirements expose the limitation of man-made knowledge as well as the state-of-the-art in knowledge system design. The second criterion seems to be very significant in AVIS. Following our experience, we found AUTORED might incidently make the wrong interpretations (false evidence) which, in terms of circuit topology, have no sound reasons to be classified as errors. For instance, AUTORED might misinterpret a bipolar transistor (BJT) as a JFET. Under most circumstances, BJT and JFET are equivalent in the sense that their interchange does not violate any circuit theories, except the difference in certain electrical properties (e.g., gain). This kind of discrepancy can only be detected by functional simulation rather than topological reasoning as is performed by AVIS. For this reason, we include the partnership with the user. It actually functions at the last stage of the verification operation. When the system displays its verification results, the designer can accept or override AVIS' suggestions through the database operators provided by the validation controller.

## 5.2 An Example

Data completeness and consistency verification are performed based on the concepts, principles, and techniques discussed so far. To illustrate the effectiveness of AVIS, let us reconsider the example mentioned in Chapter I. The circuit diagram is again shown in Figure 5.3. Obviously, after image digitization, quite a few superfluous lines, junctions are created, mainly caused poor image quality. Inevitably, several mis-recognitions and mis-interpretations are included in the AUTORED generated CAD data base. Figure 5.4 depicts a portion of the captured line file. Totally 107 line segments are created but 83 of them are either redundant or superfluous. In Figure 5.5, we show part of the generated junction file containing 51 junctions but 32 of them are superfluous or involve junction type errors. Figure 5.6 illustrates the whole set of component file. It captures 20 component symbols but 5 of them are flaws. Figure 5.7 depicts AVIS' verification operation regarding junction files; Figure 5.8 displays the overall verification statistics.

## 5.3 Interconnection Verification

The verification principles, techniques, and system described above are designed for the tasks of data completeness and data consistency verification. These two types of problems fall into the application domain of AVIS

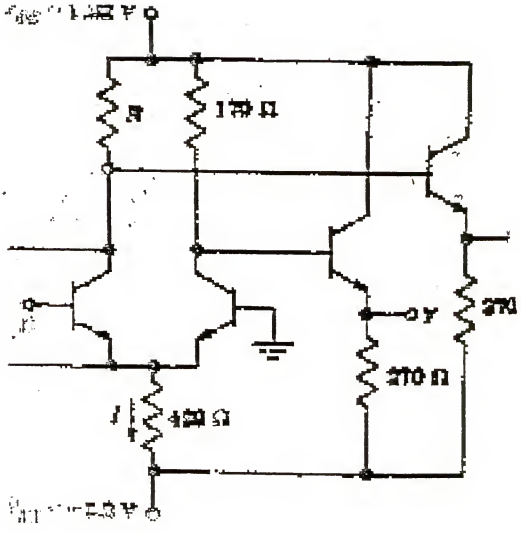



Figure 5.3 A Circuit Image after Digitization.

| X1 | Y1 | X2 | Y2 | TYPE | LENGTH | - | ORIENT |
|---|---|---|---|---|---|---|---|
| 43 | 4 | 52 | 5 | 1 | 10 | 0 | 0 |
| 15 | 8 | 23 | 8 | 0 | 9 | 0 | 0 |
| 41 | 8 | 52 | 9 | 0 | 12 | 0 | 0 |
| 65 | 9 | 70 | 9 | 1 | 6 | 0 | 0 |
| 14 | 11 | 20 | 11 | 1 | 7 | 0 | 0 |
| 50 | 26 | 196 | 28 | 0 | 147 | 0 | 0 |
| 99 | 44 | 107 | 45 | 0 | 9 | 0 | 0 |
| 112 | 44 | 117 | 44 | 1 | 6 | 0 | 0 |
| 59 | 47 | 64 | 48 | 1 | 6 | 0 | 0 |
| 100 | 48 | 107 | 48 | 1 | 8 | 0 | 0 |
| 111 | 49 | 118 | 49 | 1 | 8 | 0 | 0 |
| 58 | 51 | 63 | 51 | 1 | 6 | 0 | 0 |
| 179 | 64 | 184 | 64 | 1 | 6 | 0 | 0 |
| 48 | 72 | 133 | 74 | 0 | 86 | 0 | 0 |
| 135 | 73 | 180 | 74 | 0 | 46 | 0 | 0 |
| 186 | 86 | 191 | 86 | 1 | 6 | 0 | 0 |
| 189 | 88 | 194 | 89 | 1 | 6 | 0 | 0 |
| 139 | 97 | 145 | 97 | 1 | 7 | 0 | 0 |
| 219 | 98 | 220 | 98 | 0 | 0 | 2 | 0 |
| 192 | 100 | 216 | 101 | 0 | 25 | 0 | 0 |
| 1 | 105 | 53 | 107 | 0 | 53 | 0 | 0 |
| 83 | 105 | 140 | 107 | 0 | 58 | 0 | 0 |
| 149 | 121 | 154 | 121 | 1 | 6 | 0 | 0 |
| 201 | 127 | 215 | 128 | 0 | 15 | 0 | 0 |
| 219 | 127 | 220 | 132 | 0 | 0 | 2 | 0 |
| 19 | 129 | 36 | 130 | 0 | 18 | 0 | 0 |
| 100 | 130 | 117 | 131 | 0 | 18 | 0 | 0 |
| 177 | 131 | 182 | 131 | 1 | 6 | 0 | 0 |
| 151 | 132 | 174 | 134 | 0 | 24 | 0 | 0 |
| 36 | 138 | 41 | 138 | 1 | 6 | 0 | 0 |
| 15 | 140 | 20 | 140 | 1 | 6 | 0 | 0 |
| 88 | 140 | 94 | 141 | 1 | 7 | 0 | 0 |
| 44 | 141 | 50 | 142 | 1 | 7 | 0 | 0 |
| 86 | 143 | 91 | 143 | 1 | 6 | 0 | 0 |
| 108 | 145 | 124 | 146 | 1 | 17 | 0 | 0 |

* Number of Captured Line Symbols: 106

Figure 5.4 Part of AUTORED Generated Line File.

| X1 | Y1 | X2 | Y2 | TYPE | Xc | - | Yc |
|---|---|---|---|---|---|---|---|
| 82 | 70 | 88 | 76 | 1 | 85 | 0 | 73 |
| 151 | 70 | 157 | 76 | 1 | 154 | 0 | 73 |
| 47 | 103 | 53 | 109 | 1 | 50 | 0 | 106 |
| 47 | 70 | 53 | 76 | 2 | 50 | 0 | 73 |
| 41 | 5 | 47 | 11 | 2 | 44 | 1 | 8 |
| 81 | 23 | 87 | 29 | 2 | 84 | 1 | 26 |
| 190 | 97 | 196 | 103 | 2 | 193 | 1 | 100 |
| 152 | 24 | 158 | 30 | 2 | 155 | 1 | 27 |
| 82 | 103 | 88 | 109 | 2 | 85 | 0 | 106 |
| 97 | 127 | 103 | 133 | 2 | 100 | 0 | 130 |
| 150 | 129 | 156 | 135 | 2 | 153 | 1 | 132 |
| 47 | 152 | 53 | 158 | 2 | 50 | 1 | 155 |
| 64 | 152 | 70 | 158 | 2 | 67 | 1 | 155 |
| 79 | 175 | 85 | 181 | 2 | 82 | 1 | 178 |
| 64 | 195 | 70 | 201 | 2 | 67 | 0 | 198 |
| 150 | 196 | 156 | 202 | 2 | 153 | 1 | 199 |
| 175 | 70 | 181 | 76 | 3 | 178 | 1 | 73 |
| 47 | 5 | 53 | 11 | 3 | 50 | 1 | 8 |
| 136 | 104 | 142 | 110 | 3 | 139 | 0 | 107 |
| 172 | 154 | 178 | 160 | 3 | 175 | 1 | 157 |
| 211 | 125 | 217 | 131 | 3 | 214 | 1 | 128 |
| 33 | 127 | 39 | 133 | 3 | 36 | 0 | 130 |
| 209 | 97 | 215 | 103 | 3 | 212 | 1 | 100 |
| 82 | 175 | 88 | 181 | 4 | 85 | 0 | 178 |
| 84 | 24 | 90 | 30 | 4 | 87 | 0 | 27 |
| 113 | 142 | 119 | 148 | 4 | 116 | 0 | 145 |
| 47 | 1 | 53 | 7 | 5 | 50 | 1 | 4 |
| 98 | 41 | 104 | 47 | 5 | 101 | 1 | 44 |
| 101 | 41 | 107 | 47 | 5 | 104 | 0 | 44 |
| 63 | 23 | 69 | 29 | 5 | 66 | 0 | 26 |
| 66 | 23 | 72 | 29 | 5 | 69 | 0 | 26 |
| 162 | 153 | 168 | 159 | 5 | 165 | 1 | 156 |
| 166 | 153 | 172 | 159 | 5 | 169 | 1 | 156 |
| 169 | 154 | 175 | 160 | 5 | 172 | 0 | 157 |
| 201 | 124 | 207 | 130 | 5 | 204 | 0 | 127 |

* Number of Captured Junction Symbols: 51

Figure 5.5 Part of AUTORED Generated Junction File.

| X1 | Y1 | X2 | Y2 | - | ORIENT | TYPE |
|---|---|---|---|---|---|---|
| 64 | 4 | 71 | 11 | 0 | 0 | 8275 |
| 80 | 32 | 91 | 65 | 0 | 2 | 8261 |
| 45 | 33 | 56 | 65 | 0 | 2 | 8261 |
| 177 | 56 | 195 | 91 | 0 | 1 | 8258 |
| 138 | 89 | 156 | 124 | 0 | 1 | 8258 |
| 210 | 96 | 217 | 103 | 0 | 0 | 8275 |
| 34 | 113 | 51 | 146 | 0 | 1 | 8258 |
| 84 | 114 | 102 | 146 | 0 | 3 | 8258 |
| 188 | 115 | 199 | 146 | 0 | 2 | 8261 |
| 200 | 125 | 206 | 134 | 0 | 0 | 8275 |
| 207 | 125 | 216 | 133 | 0 | 1 | 8281 |
| 14 | 126 | 21 | 133 | 0 | 0 | 8275 |
| 169 | 129 | 175 | 136 | 0 | 0 | 8275 |
| 149 | 141 | 159 | 173 | 0 | 2 | 8261 |
| 171 | 154 | 178 | 163 | 0 | 0 | 8275 |
| 62 | 155 | 72 | 192 | 0 | 2 | 8261 |
| 161 | 156 | 168 | 163 | 0 | 0 | 8275 |
| 7 | 210 | 15 | 220 | 0 | 2 | 8261 |
| 64 | 213 | 71 | 220 | 0 | 0 | 8275 |
| 108 | 142 | 124 | 157 | 0 | 0 | 8263 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 999 | 999 | 999 | 999 | 999 | 999 | 999 |

FORTRAN STOP

* Number of Captured Component Symbols: 20

Figure 5.6 AUTORED Generated Component File

```
 THE CLUSTERED LEAF NODE IS NODE#       21
  THE IDENTIFIED DEFECT IS  J1.1.4
 ~~~~~~~~~~~~~~~~~~~~~~~~

  JUNCTION #        1 IS NORMAL !

 ~~~~~~~~~~~~~~~~~~~~~~~~
 THE CLUSTERED LEAF NODE IS NODE#       21
  THE IDENTIFIED DEFECT IS  J1.1.4
 ~~~~~~~~~~~~~~~~~~~~~~~~

  JUNCTION #        2 IS NORMAL !

 ~~~~~~~~~~~~~~~~~~~~~~~~
 THE CLUSTERED LEAF NODE IS NODE#       49
  THE IDENTIFIED DEFECT IS  J1.3.1.4
 ++++++++++++++++++++++++

  THE DIAGNOSED RESULT OF JUNCTION#       3    IS:

TYPE-1 JUNCTION ERROR

DEFECTCODE=      26
TREATCODE=       16

  THE SUGGESTED TREATMENT IS TO ( CORRECT TYPE-1 JUNCTION TYPE ERROR

  THE PLANNED ACTION IS ( GROUP-1 JUNCTION TYPE ERROR CORRECTION    )

++++++++++++++++++++++++
THE CLUSTERED LEAF NODE IS NODE#       50
 THE IDENTIFIED DEFECT IS
++++++++++++++++++++++++

  THE DIAGNOSED RESULT OF JUNCTION#       4    IS:

TYPE-2 JUNCTION ERROR

DEFECTCODE=      27
TREATCODE=       17

  THE SUGGESTED TREATMENT IS TO ( CORRECT TYPE-2 JUNCTION TYPE ERROR

  THE PLANNED ACTION IS ( GROUP-1 JUNCTION TYPE ERROR CORRECTION    )
```

Figure 5.7 Programming Aspects in Junction File Verification

OVERALL STATISTICS OF THE CIRCUIT FROM AUTORED OUTPUT

TOTAL NUMBER OF LINES: 106
TOTAL NUMBER OF JUNCTIONS: 51
TOTAL NUMBER OF COMPONENTS: 20

DIAGNOSIS OF AUTORED OUTPUT

51 DENOTATION-LINE SYMBOLS WERE REMOVED:

1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 19, 25, 28, 31
39, 40, 41, 44, 45, 47, 49, 50, 51, 52, 54, 55
56, 57, 61, 62, 63, 64, 65, 66, 68, 70, 71, 75, 76
77, 78, 80, 81, 82, 83, 85, 90, 91, 95, 97, 98

26 COMPONENT-BOUNDARY-LINE SYMBOLS WERE REMOVED:

4, 13, 16, 17, 18, 23, 30, 32, 33, 34, 35, 36, 46, 48
53, 78, 86, 92, 93, 94, 96, 102, 103, 104, 105, 106

2 BROKEN LINES WERE IDENTIFIED:

14, 69

1 SUPERFLUOUS LINE SYMBOL WAS REMOVED

43

13 DENOTATION-JUNCTION SYMBOLS WERE REMOVED:

6, 14, 24, 25, 27, 28, 29, 30, 31, 36, 37, 41, 46

10 COMPONENT-JUNCTION SYMBOLS WERE REMOVED:

10, 16, 18, 19, 20, 21, 22, 26, 34, 48

11 SUPERFLUOUS JUNCTION WERE REMOVED:

5, 18, 32, 33, 35, 38, 39, 40, 42, 44, 50

5 SUPERFLUOUS COMPONENTS WERE REMOVED:

10, 11, 15, 17, 18

Figure 5.8 Verification Statistics

so far discussed. However, for the third task, interconnection verification, we make use of a different technique simply because interconnection is a property with little sense of classification. Because CAD design for digital systems has its special methodology (such as: silicon compilers usually start the design process from specification description rather than the circuit sketches) our interconnection verification, therefore, concentrates on analog circuits only, which is also the domain AUTORED is designed for. Two objectives are sought under this verification performance:

(1) Structural Error--referring to the inadequate or hazardous connections which hinder the circuit from functioning properly, such as open circuit, grounding the voltage source, mis-connecting the output port to a constant signal source, etc. Figure 5.9 shows some of these aspects.

(2) Behavioral Error--regarding the violation of the operating conditions needed for the devices to function normally, such as blocking the DC biasing currents from transistors, permanently shutting off devices due to inadequate reverse bias, etc. Figure 5.9 also depicts some examples in this category.

### 5.3.1 Structural Error Verification

This type of error is detected following the technique of net list check (Begg, 1984). Two major types of problems

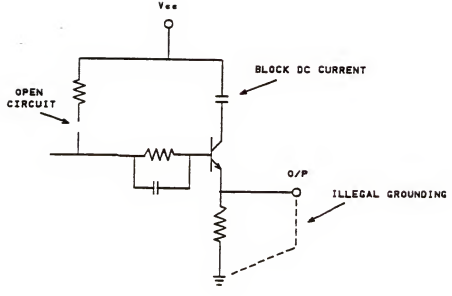



Figure 5.9 Examples Regarding Structural Errors and Behavioral Errors

are categorized including (1) open circuit, and (2) improper connection. For the open circuit detection, we count the connection number of each component and compare that with the default value created from the device model. Any difference signifies the possibility of an open circuit. The second case, improper connection, is detected by checking the component type at the connecting nets. Figure 5.10 illustrates these two verification operations.

### 5.3.2 Behavioral Error Verification

Two verification operators, point verifier and path verifier, are employed behavioral error verification. Incorporating the technique of excitation state labeling as well as the Kirchhoff's Current Law (KCL), these two operators specify the necessary conditions of the propriety of interconnection. The whole verification framework is constructed on the basis of the network model defined as the following.

Network model. The network model for interconnection verification consists of a set of nodes $\{n_1,..,n_p\}$ linked together by a set of branches $\{b_1,..,b_q\}$ (Bryant, 1984). Each node is classified as either a source node, component node, or junction net node. Branches are the symbols denoting the connectivity between nodes. Each node has a certain number of connecting branches, termed node branches. The number of node branches is determined by node type as well as network

(A) OPEN CIRCUIT DETECTION

| Component Type | No. of Default Connection | No. of Net Connection | |
|---|---|---|---|
| R1 | 2 | 2 | |
| BJT-1 | 3 or 2 | 2 | |
| C2 | 2 | 2 | |
| D1 | 2 | 2 | |
| R2 | 2 | 1 | <——— Open Circuit Detected |
| T1(I/P) | 1 | 1 | |
| T2(O/P) | 1 | 1 | |

(B) DETECTING IMPROPER CONNECTION

| NET-1 | NET-2 | NET-3 | |
|---|---|---|---|
| R1-1 | V1-1 | V2-1 | <— Output Port |
| R2-2 | R5-1 | R3-2 | <— Connects to |
| C1-2 | BJT-1-3 | T2(O/P) | <— Voltage |
| BJT-2-1 | R2-1 | -- | |
| -- | -- | -- | |
| -- | -- | | |

Figure 5.10 Structural Errors Detection

configuration. Each node branch $b_i$ has a state, denoted $s(b_i)$, in the set {0, 1, X} representing the current flowing direction relative to the connected node (0: inlet current, 1: outlet, X: undefined). We define the state of a node as the combination of its node branch states. Source nodes including voltage source, ground port, as well as input ports, provide strong signal to the system and their states are prespecified regardless of the actions of the network. Certain component nodes such as transistor or diode nodes, also have their states predetermined by their behavioral models. Junction net nodes and the component nodes denoting passive devices, such as resistors or capacitors, have their states totally determined by their adjacent connections. The output port in our model (same as AUTORED) is considered as a component node with its state always decided by the network. Figure 5.11 shows some of the conventions just specified.

Node excitation. A node is exciting when one of its node branches has its state determined. The excitation state of a node is defined as the combination of the excitation states of its connecting node branches. Depending on the node type, the node excitation states might be prespecified (e.g., source nodes), or determined by the excitation states of its constituent branches, such as junction net nodes or component nodes representing passive devices (e.g., resistor). In the network model shown in Figure 5.12 the excitation states

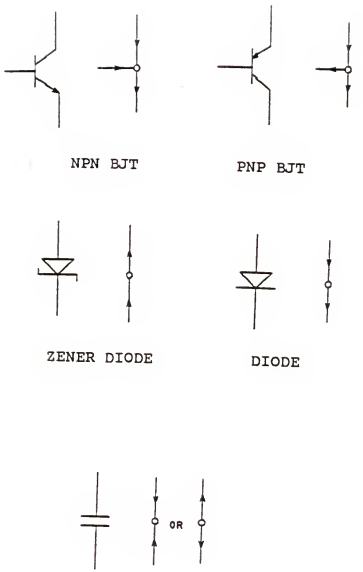
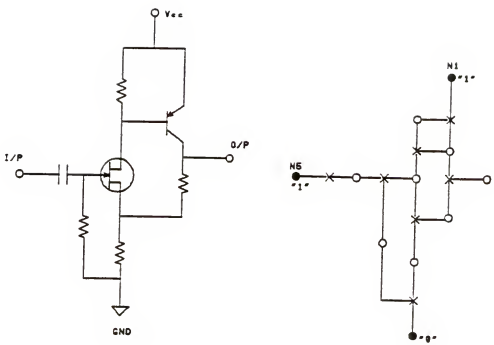



Figure 5.11 Some Component Nodes with Their Pre-specified Node Excitation States




– : branch
● : source node
o : component node
× : junction net node

Figure 5.12 A Transistor Circuit with Its Network Model

of node#1 is predefined as +1. For node#5, it is either +1 or 0 depending on the input signal polarity. Similarly, for node#9 denoting a ground port, the state is 0 characterizing its property as a power sink.

Branch excitation. A branch is defined as excited when its state is determined. Since each branch has at least one connection to the node, the branch excitation state generally is a function of its connecting nodes. The propagating of state excitation can be created along the network shown in this notation:

$$S(b_p(n_q(b_{q1}, .., b_{qr})))$$

It means that the excitation state of branch $b_p$ is a function of the excitation state of node $n_q$ which in turn is a function of the excitation states of its node branches $(b_{q1}, .., b_q)$. This enables the branches to broadcast excitation information from node to node and finally determine the excitation state of the whole network.

Network excitation. The excitation state of a network is defined as the combination of the excitation states of the constituent nodes and branches with source nodes being held fixed in a preset state (e.g., 1). The network excitation, if it can be found, has the meaning of DC current flowing in the direction of the steady-state of the network, which enables the kind of one-step deduction (Sussman and Stallman, 1975)

as often employed by the experienced circuit experts to "analyze" circuits visually. This technique can not explore the complete electrical properties of the circuit, however, it is good enough for verifying the adequacy of interconnection.

KCL constraint in network excitation. KCL says that "For any lumped electric circuit, for any of its nodes, and at any time, the algebraic sum of all the branch currents leaving the node is zero." (Desoer and Kuh, 1969). The reason behind this law is the conservation principle which also implies the aspect of current continuity. KCL is often used in circuit analysis. Parallel to the quantitative description, we have the following statement regarding its qualitative property.

Proposition 5.1 "For any network, for any of its nodes, and at any time, the inverted E-EXCLUSIVE-OR of all branch excitation states of the node is zero," where the OR operator covering ternary logic involving multiple operands is defined as:

$$Y = \cup A_i = \begin{cases} 1 & \text{if all } A_i=1, \\ X & \text{if some } A_i=X, \\ 0 & \text{otherwise.} \end{cases}$$

The function of E-EXCLUSIVE-OR (extended exclusive-or) is

defined as:

$$Y = \oplus A_i = \begin{cases} 0 & \text{if all } A_i=0 \quad \text{or} \quad \text{all } A_i=1 \\ 1 & \text{otherwise} \end{cases}$$

with the following truth table:

| A1 | A2 | .... | Ai | ... | An | Y | $\overline{Y}$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | 0 | | 0 | 0 | 1 |
| 1 | 0 | | 0 | | 0 | 1 | 0 |
| X | 0 | | 0 | | 0 | 1 | 0 |
| .. (other combinations).. | | | | | | 1 | 0 |
| 1 | 1 | .... | 1 | ... | 1 | 0 | 1 |

Proof: From KCL, the branch currents at any node must satisfy

$$I_{b1} + I_{b2} + I_{b3} + .... + I_{bn} = 0 \qquad (5.1)$$

(5.1) can be reorganized into

$$(I_{b1'}+I_{b2'}+ .... +I_{bn'}) + (I_{b1''}+I_{b2''}+ .... +I_{bn''}) = 0 \qquad (5.2)$$

where the element in each parenthesis has the same polarity. Equation (5.2) can be expressed into the form:

$$(I_{b1'}+I_{b2'}+ .... +I_{bn'}) = - (I_{b1''}+I_{b2''}+ .... +I_{bn''}) \qquad (5.3)$$

Without lost the generality, if the left hand side denotes the inlet current, then the right hand side denotes the outlet current. In other words, KCL qualitatively demands that at any

node, the inlet current and outlet current must balance. Referring to the notations of node branch excitation state, we have the conclusion that not all the node branches can have the same excitation states. This conclusion is reflected at the top and the bottom rows in the truth table of the E-EXCLUSIVE-OR, and thus we prove this proposition.

Figure 5.13 (a) shows two examples regarding node excitation states, where node#a, following this proposition, is acceptable, but node#b is not.

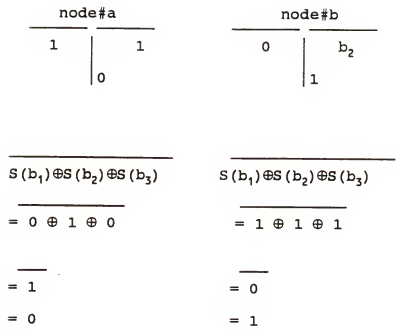
An important application of this proposition is to predict the legitimate excitation state of an unexcited branch as summarized in the following proposition:

Proposition 5.2 The excitation state of the node branch can be specified by the following relations:

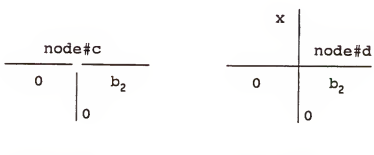
$$S(b_j) \begin{cases} = 1 \text{ ; if all } b_i = 0 \text{ , } i = 1,.., n, \ i \neq j \\ = 0 \text{ ; if all } b_i = 1 \text{ , } i = 1,.., n, \ i \neq j \\ = x \text{ ; otherwise} \end{cases}$$

where n is the number of overall node branches, $b_j$ is the node branch with unknown state.

As an illustration, the node#c in Figure 5.13 (b) has only one of its three branches with undecided excitation state. Following this proposition it can be labeled as 1.




(a) Using KCL as Verification Rule.




(b) The Branch Excitation State of $b_2$ in node#c can be labeled following Proposition 5.2.

Figure 5.13 Node Branch Excitation State Determination.

5.3.3 Labeling Network Excitation State.

Definition 5.1 Excitation labeling is a set of nodes and branches denoted as (Initiator, Follower, Terminator), where "Initiator" can be either source nodes or component nodes representing the active devices. "Follower" contains the set of state-excited elements including nodes and branches. "Terminator" covers the set of elements, either source nodes, component nodes, or branches. Within this region all the node excitation states except the terminators are determined.

Since only a partial information in terms of qualitative variables is actually utilized, the labeling of the network excitation state proceeds based on the following rules:

Labeling Rule-1: The excitation state of positive voltage source is set to "1", and "0" for negative voltage source.

Labeling Rule-2: The excitation state of ground node can be either "0" (relative to the positive voltage source) or "1" (relative to the negative voltage source).

Labeling Rule-3: All the node branches (except the one with direct connection) of the junction node connecting to a positive voltage source have excitation state "1", and "0" for connecting to a negative voltage source.

Labeling Rule-4: No internal loop is allowed inside the network.

Some realistic sense of these four labeling rules are shown in Figure 5.14.

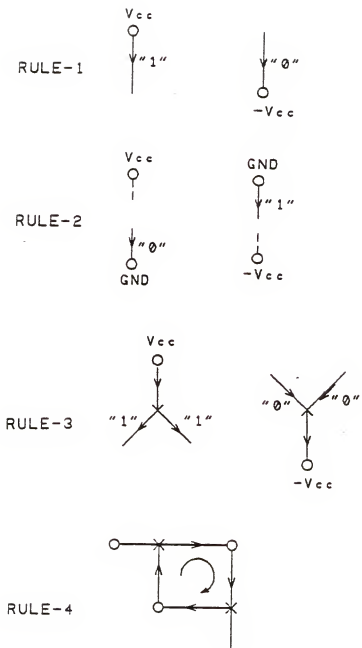



Figure 5.14 Rules for Labeling Network Excitation State

Initiator precedence. Since more than one node may qualify as an initiator, a precedence order is also set up to regulate the priority:

Precedence Rule-1: SOURCE NODE > COMPONENT NODE

Precedence Rule-2: (inside the set of source nodes)

+VOLTAGE SOURCE > -VOLTAGE SOURCE
> SIGNAL INPUT PORT > GROUND

Precedence Rule-3: (inside the set of component nodes)

3-PORT ACTIVE DEVICE > 2-PORT ACTIVE DEVICE

These rules are created based on the general circuit theories.

Termination. Following the labeling and precedence rules in addition to proposition 5.2, the labeling process starts from the initiator and spreads out as far as possible. However, the node excitation state, according to proposition 5.2, might be undecided (x), hence, the propagation of the labeling process terminates usually under the following two conditions:

(1) Encountering a junction net node with the number of unexcited branches greater than one so that Proposition 5.2 can not specify the excitation state of the unknown branch.

(2) Meet with the source nodes or component nodes with prespecified excitation states.

The region after the termination of a specific labeling process is termed an excited area, and the set of its terminators forms the boundary of that excited area. A "relay" mechanism is implemented. It allows a new labeling process to take place after the termination of the current process. The excitation state of a network after all the relayed labeling processes might be the union of a set of excited areas and an unexcited area which is defined as the union of the network fragments with the excitation state undetermined. Figure 5.15 symbolically illustrates this aspect.

## 5.4 Verification Operators

Two operators: point verifier and path verifier, are employed for identifying the possible behavioral errors after the labeling process.

### 5.4.1 Point Verifier

Point verifier is a verification operator with the capability of detecting any violation of the junction excitation states against Proposition 5.1. Its diagnosis

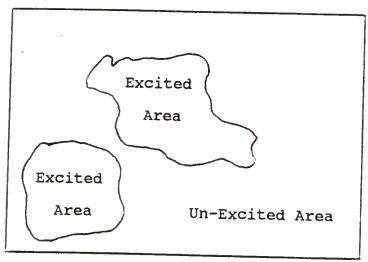



Figure 5.15 The Network Excitation State Is the Union of Several Excited Areas and An Un-Excited Area.

principle is based on the following rule:

<⊕ , b(n#i), 0> =====> <STATUS, n#i, ERROR >

It specifies that the node #i is erroneous if the E-Exclusive-OR of its node branch excitation states equals to 0. This rule is derived from Proposition 5.1.

A limitation of this point verifier is the case when the diagnosed junction node involves node branches with undefined states, which always make the E-Exclusive-OR value equal to "x", thus, pass this check rule. This is compensated by the path verifier.

5.4.2 Path Verifier

For an active device (e.g., transistor) to function properly, DC biasing current is required. The DC current, following the device model, is a function of the input driving current (e.g., base current for BJT) and quite often (i.e., via design convention) it is directly fed from voltage source through a simple protection mechanism (e.g., a current-limiting resistor) to the active device (either the collector of NPN transistor or the emitter of PNP transistor). Based on this observation, we have the following heuristic:

"THE ROUTE OF DC BIASING CURRENT IS THE SHORTEST PATH FROM THE VOLTAGE SOURCE TO THE ACTIVE DEVICE."

Unlike the check rule utilized by the point verifier, this heuristic is obtained from design conventions and device models is not rigorous enough to be a law, because we can use some tricky circuitry to violate this heuristic (e.g., not shortest path) and still make the circuit work. But it seems to hold well on most circuits and provides good indication about the hidden errors. For instance in the circuit diagram shown in Figure 5.9, this heuristic can detect the behavioral error caused by the capacitor which blocks the DC biasing current from the transistor. We define the route of DC biasing current as the feed path. Specifically, three types of feed paths can be classified:

Type-1 Feed Path: A path consists of a set of connecting branches from the source node representing positive voltage source, via a shortest length, through the component nodes representing active devices, to the source node representing ground (e.g., from +V to the collector of an NPN BJT, then to the emitter of the NPN BJT, then to ground).

Type-2 Feed Path: A path consists of a set of connecting branches from the source node representing ground, via a shortest length, through the component nodes representing active devices, to the source node representing negative voltage source (e.g., from Ground to the emitter of an PNP BJT, then to the collector of the PNP BJT, then to -V).

Type-3 Feed Path: A path consists of a set of connecting branches from the source node representing the positive voltage source, via a shortest length, through the component nodes representing active devices, to the source node representing negative voltage source (e.g., from +V to the collector of the NPN BJT, then to the emitter of the NPN BJT, then to -V).

In those feed paths, only the node branches emitting from nodes to the connecting nodes are considered.

Based on this heuristic rule, the path verifier takes the following two actions:

(1) Identify the feed path for each active device. Here the feed path can be identified following the algorithms of tree finding developed in graph theory (Deo, 1974).

(2) Check the combination of branch excitation states along this feed path and localize any possible behavioral errors. An operator, SGN, is devised for this purpose. Its function is to count the repetition of state change (eg. 1 --> 0, or 0 -->1) between any two consecutive digits in a binary number string, i.e.,

$$m = SGN \{ P(n) \}$$

where P(n): a binary number string (e.g., the sequence of the branch excitation states along the feed path).

m: the overall counts of state change.

Since the feed path contains only those branches emitting from

nodes, P(n) represents the flowing sense of the DC biasing current. The nonzero value of SGN{P(n)} implies the DC biasing current is not continuous, therefore, there is a great possibility of behavioral error.

Following this heuristic just mentioned, the verification principle of the path verifier is:

Path verification principle-- A necessary condition for a feed path to be topologically sound is that the value of SGN {P(n)} must be zero. Any pair of sign changes represents the existence of a behavioral error.

Since each element $b_i$ of P(n) is in the set {0, 1, X}. two operation rules are included to deal with the ternary logic:

Merging Rule:

$$b_i \times x \ldots \times b_j = b_i \times b_j$$

Absorption Rule:

$$b_i \times b_j = b_i \; b_j$$

where $b_i$, $b_j \in \{0, 1\}$.

These two rules practically suggest that in a normal design the DC biasing current tends to be continuous. This suggestion is reasonable since the undefined excitation state X can occur, following proposition 5.2, only when the cardinality of a node is greater than three. That means we

have more freedom and possibility to arrange a correct feed path.

Example. Consider the example shown in Figure 5.16. Inside this transistor circuit (partial view only), there is a behavioral error caused by diode D1, which virtually blocks the DC operating current of transistor T1. It might happen because of poor design, or mis-classification of a zener diode as a diode by AUTORED. The verification process begins with the labeling of the network excitation state. According to the arbitration rule-1, node#1 is first selected as the "initiator" and starts the labeling process. The resultant network excitation state with arrow head denoting the flowing sense of DC current is shown in Figure 5.16(b). The point verifier then checks every junction net node and identifies node#10 and node#12 as erroneous since these two nodes have the E-Exclusive-OR of their node branch states equal to 0. Moreover, the path verifier identifies the feed path of T1 as:

$$P(b) = b1\ b2\ b4\ b6\ b8\ b10\ b11\ b12\ b13\ b14$$
$$= (1111100111)$$

and the number of status changes is

$$SGN\ \{P(b)\}=2.$$

Hence, the path verifier suggests that one behavioral error occurred between node#10 and node#12.

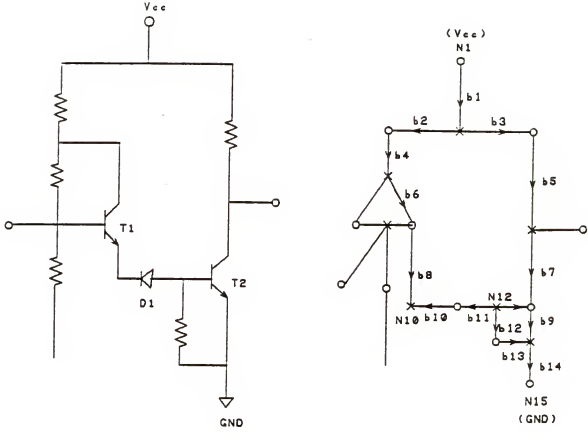



Figure 5.16 A Transistor Circuit with Improper Diode Polarity.

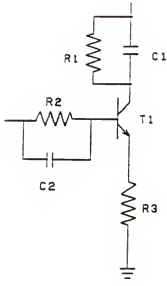
For T2, the feed path is

$$P(b) = b1\ b3\ b5\ b7\ b9\ b14$$
$$= (111111)$$

and $SGN\{P(b)\}=0,$

so no error is declared.

Alternate Path. As mentioned earlier, in practical circuit design, the DC biasing current might flow through a protection mechanism or other arrangement, therefore, the feed path, under this circumstance, might not be the shortest path as originally defined. From the verification standpoint, feed path is only an indication of the route providing the biasing current for active devices. Once there exist certain paths which create the same effect, the active devices are still able to function normally. Concerning this realistic situation, when a component node in the feed path is first identified as improper, we try to find a substitute, termed alternate path, which, replacing the inadequate component node, creates another branch for the DC current. Alternate path is defined as the path in parallel to the "improper" component node without going through any source nodes and it has no overlap with the original feed path. Figure 5.17 illustrates the finding of alternate path and the corresponding adjustment of feed path.




ALTERNATE PATH:
(1) IN PARALLEL CONNECTION
(2) NO VOLTAGE SOURCE
(3) NO OVERLAP WITH ORIGINAL FEED PATH

Figure 5.17 Finding Alternate Path for DC Biasing Current

Remark. Since network excitation state (i.e., steady-state response) is only a partial description of the circuit, no electrical properties of quantitative type are provided based on the interconnection verification process just elaborated. On the contrary, they should be calculated from simulation. Our verification objective is to ferret out those erroneous factors hidden inside the circuits such that the designer can revise his/her prototype design before moving into the expensive simulation process. This verification aspect, hence, is a complement to the traditional simulation or the quantitative type circuit analysis. However it also has its special capability, for instance the detection of the behavioral error illustrated in the last example. It might be difficult to find that discrepancy based on simulation routines which usually end up with some output data that greatly deviate from our expectation without any clear indications of trouble sources. Since behavioral errors usually relate to certain design intentions of the user, we do not intend to make any validation suggestions. Instead, the detected behavioral errors are fed back to the designer for final identification through the user interface.

## 5.5 Conclusion

In this chapter, we have presented the whole integrated AVIS system. The blueprint mainly follows the framework

elaborated in the last few chapters with the inclusion of validation controller, and some other design aspects, such as the multi-pass diagnosis strategy to reduce the effect of the nonmonotonicity in the problem domain. We also present a method for interconnection verification which is one of AVIS key design objectives. Two verification operators, point verifier and path verifier are devised for this purpose. Incorporating the technique of network excitation state labeling as well as the Kirchhoff's Current Law (KCL), these two operators are able to justify the topological propriety of circuits. The AVIS-MK1 system has been implemented in a VAX11/750 mini-computer. This project successfully demonstrates the feasibility of integrating machine intelligence with CAD/CAM technologies. All programs used in this work are available as text files on a floppy disk by writing to:

Center for Information Research
E314 CSE Bldg.
University of Florida
Gainesville, FL 32611

# CHAPTER VI
# CONCLUSION

## 6.1 Summary

It is all known that data entry by human operators is time-consuming and error-prone. Automated machines for reading and interpreting schematic diagrams are needed for enhancing the current CAD capabilities, and the Center for Information Research has developed AUTORED, an automatic schematic reading system, to response this need (Tou and Cheng, 1983a). AVIS is the sequel of AUTORED with the objective of verifying the AUTORED-generated CAD database before it can be used in design automation. AVIS is a knowledge-based system. Concerning the special properties in the problem domain, the design of AVIS, instead of following the most publicized rule-based approach, utilizes the data-driven, pattern-directed inference technique, and draws the design blueprint from PADIKS. PADIKS, the acronym for pattern-directed expert system, generates its strength from a structured knowledge base in conjunction with pattern recognition techniques for efficient knowledge seeking and inference. For the AVIS project, we further consolidate this design concept into a full-fledged framework including the pattern-based knowledge rarefaction and representation, the sketch-details scheme for knowledge base configuration,

an adaptive inference control, an Non-Bayesian method for approximate reasoning, etc. The main achievements of this research are summarized as follows:

(1) A pattern-based structured representation for knowledge has been elaborated in Chapter II. Unlike semantic networks, frames, or other formalisms, domain knowledge is represented in terms of three primitives: entities, attributes, and relationships. Entities are types of information which are tied together by a set of relations to form a knowledge base. Attributes are characteristics of other knowledge entities. Relationships semantically organize the knowledge entities into an associative tree with multiple levels of details. The associative tree structure, in fact, realizes the hierarchical clustering of knowledge entities, which not only coincides with our intuition about how the domain is structured but also facilitates efficient breaking down of generic concepts into the fine grain size we care to explicate. A knowledge base configuration reflecting this representation concept is also proposed for AVIS. It contains three major parts: knowledge sketch, knowledge details, and context generator. The knowledge sketch is the strategic outline of domain knowledge, while the knowledge details form the supporting facility of the knowledge sketch. A context generator consisting of a number of procedures, on the other hand, implements AVIS'

self-sensing capabilities. This architecture materially integrates symbolic reasoning with computation accuracy.

(2) As far as the design of the structured knowledge base is concerned, we propose that the knowledge domain be classified into two types: well-organized and semi-organized. For the former case where the domain has crisp semantics and structure, the issue can be formulated as a knowledge base generation problem. When domain structure is usually fuzzy, we propose that the knowledge domain be modelled by an N-Cube, and the knowledge sketch be generated based on the technique of N-Cube partition. The determination of partition rule can be subjective in accordance with the expert's knowledge, or objective in terms of certain performance indices, such as minimizing computation efficiency or maximizing concept coherence. A discussion about this subject is included in the APPENDIX.

(3) The most beneficial advantage of knowledge structure can provide is the strategic partition and distribution of knowledge entities into an associative tree structure, which during inference virtually limits the selection of hypotheses from a huge body of candidates into a few choices. This benefit is realized in SICS (short for squeeze inference control strategy), the inference control metaphor we propose for AVIS. SICS is a beam search with the capability of adaptively adjusting its

beam-width based on the appearance of diagnostic contexts. It decomposes the inference mechanism into two major steps: integral analysis and differential diagnosis. The mission in the integral analysis is to create an upper bound of the plausible defect instances from the equivalence class of the relations defined by the diagnostic contexts. The differential diagnosis, on the other hand, tightens the boundary into the best upper bound, and manages (along with a termination check routine) to create the best lower bound. Therefore, SICS terminates the inference process when these two bounds become equal. The theoretic framework is derived from the rough sets theory. The achieved solution description, conditioned on the knowledge structure and the observability of attributes, can be proved to be complete and concise; thus in theory, back-tracking is not needed. Another significant advantage of SICS regards the effectiveness in dealing with multi-faults diagnosis. In multi-faults diagnosis, the computation resources versus a large number of active hypotheses and their combination is a difficult issue for all kinds of sequential control strategies. However, due to the clustering capability of SICS, inference proceeds via a pseudoparallel control scheme.

(4) We further expand SICS' capability to cover approximate reasoning. In accordance with the inference logic of SICS, we propose a Non-Bayesian framework based on three foundation concepts: evidence space, belief characteristic function (BCF), and uncertain pattern matching. Evidence space facilitates an intuitive but effective means for reporting uncertain or contradictory information. BCF is an operator mirroring the human's mental favoritism and betting character of how to lessen the ambiguity reported in the evidence space. Uncertain pattern matching, on the other hand, is the rule for evidence combination. Via these three uncertainty aspects, we are able to embed the human subjectivity into the decision making process, which, as believed by the Non-Bayesian people, is a knowledge-intensive task; the domain experts have their own heuristic for dealing with typical kind of uncertainty (Chandrasekaran and Tanner, 1986). From this base the approximate reasoning in SICS is formulated into an uncertainty calculation cycle covering three conceptual levels: attribute quality, descriptor realization quality, and hypothesis confirmation degree. The approximate reasoning can be carried out uniformly in the sense that the three foundation concepts, evidence space, BCF function, and uncertain pattern matching, can be applied, alternatively, at those three different conceptual levels

in a coherent way.

(5) AVIS, targeted at the AUTORED generated CAD data base, is implemented. It purses three verification performances including data completeness check, data consistency check, and interconnection verification. The first two objectives, following the inherent taxonomic properties in the data records, are formulated into the classification problem with the design concepts drawn from PADIKS as interpreted in Chapter II, III, and IV. The third system objective, due to its little sense of classification, is accomplished via a different approach. An excitation model is used to label, in a qualitative sense, the steady state response of the network. Two verification operators, point verifier and path verifier, are devised to conduct the interconnection verification concerning structural error and behavioral error, two major types of discrepancies. The diagnostic principles are derived from Kirchhoff's Current Law in conjunction with certain circuit theories, electrical properties, design conventions, and device models. AVIS together with AUTORED is able to form a powerful schematic data entry processor for CAD machines. This project also demonstrates the feasibility of integrating machine intelligence with CAD/CAM technologies.

## 6.2 Areas for Future Work

In this dissertation, we have proposed some principles and techniques for knowledge system design, and also a demonstration system, AVIS, for CAD data base verification. Although, the results are encouraging, it is by no means complete. More research work is needed to expand the design concepts as well as the functions of AVIS. In the specific problem areas that we have addressed here, we recommend that further research be carried out on the following directions:

(1) Although the organization of knowledge base depicted in Chapter II is typical but only applicable to the domains equipped with classificatory property. It is possible to extend this concept to cover other features, such as casual relationship and temporal logic, which are also prominent in the real world affairs. For this concern, we suggest of using a multi-paged knowledge structure with each page hosting a unique characteristic aspect of the domain. Obviously a more intelligent inference strategy will be needed to manipulate a multi-paged knowledge base containing much more complicated functionalities and interactions.

(2) A significant feature in the framework of approximate reasoning illustrated in Chapter IV is the representation of the human's betting character by virtue of Belief Characteristic Function (BCF) for pooling together conflicting evidence. How to explore and categorize more

BCF functions denoting the typical favoritism of people is yet an unsolved problem. Moreover, the possibility of extending the idea of BCF to cover aspects of data filtering employed in Bayesian systems is also unexplored. Further research on these subjects should be valuable.

(3) In the current version of AVIS, the evidential weightings $ew_+$, $ew_-$, are obtained by thresholding the physical values of attributes into binary states. An improved scheme might be to calculate these weights based on fuzzy membership concept, so that the distribution of $ew_+$ and $ew_-$ could be multi-valued. For this purpose, the statistics about the distribution of attribute values are needed for generating the proper membership functions.

(4) The AVIS project, at the present time, focuses on the issue of verifying the AUTORED-captured CAD data base. The same idea certainly can be applied to other kinds of data bases. One profitable but challenging research direction is the analysis of simulation results. In pursuing this objective, a new version of AVIS with more powerful computational knowledge will be needed.

After several years of research in the field of expert systems, we come to believe that there exist solutions to the problems mentioned above. We hope that more researchers will take up these challenging problems in an effort to develop a more versatile AVIS system for CAD/design automation, and we

hope that this dissertation can contribute to the realization of those objectives in a short time.

## APPENDIX
## KNOWLEDGE SKETCH GENERATION

In terms of knowledge structure the real world domains can be classified into two categories: (1) well-organized, and (2) semi-organized (Dai and Tou, 1988). The former refers to those domains having long been studied with the domain structure having traditionally been formulated and documented. Medical knowledge, biological knowledge, and several others scientific fields all belong to this category. Contrarily, the latter class refers to the domains which are either poorly-understood or loosely-structured. Individual concepts or objects might be addressed adequately but the overall semantic relations are fuzzy or even unknown. For the well-organized case, since the taxonomic relationships have already been understood, the generation of knowledge sketch can be systematic. The APRIKS system has an excellent discussion about this subject (Tou and Cheng, 1983b). For the semi-organized case, the problem may be solved based on N-Cube model and the technique of n-cube partition.

A.1 Knowledge Domain

A knowledge domain, $\Phi$, representing the a priori knowledge, is expressed as the quadruplet:

$$\Phi = < E, \Omega, \Gamma, \mathring{A} >$$

where E is the entity space consisting a set of knowledge entities,

$\Omega$ is the set of attribute primitives,

$\Gamma = \{w_i\}$, $w_i \epsilon \Omega$, is the set of pattern vectors,

$\mathring{A}$, is the set of mappings from E to the power set of $\Omega$. Knowledge entity, e, $e \epsilon E$, is the phenomenon of interest in the knowledge domain. In practice, it might represent a subject, goal, idea, opinion, or other substantial, or conceptual items. The entity space E, containing the whole set of entities, delimits the boundary of domain knowledge. We define the primitive entities as the entities representing the ultimate conceptual instances in the knowledge domain.

Attribute primitive w, $w \epsilon \Omega$, is the primitive feature adequate in characterizing the practical meaning of entity. The entire set of attribute primitives forms the comprehensive set of vocabulary by which any entity could be properly described. Attribute primitive can be single-valued or multi-valued, numerical or symbolic. In general, symbolic attribute can be further categorized into two types, nominal and structured (Gordon, 1981). For the former case, elements of attribute variables are independent, while for the latter case superiority is embedded to certain elements over the others.

Here we consider only the attribute primitives of nominal type, and it is assumed that the embedded authority of structural attributes could be properly decomposed and distributed such that it can be expressed as a simple conjunction or disjunction of nominal attributes.

Any collection of attribute instances is called a pattern vector. Some pattern vectors have realistic sense in characterizing the entities. Some are just an arbitrary union of attributes without meaningful indications.

$\alpha \epsilon \mathring{A}$, is a mapping relation. For each knowledge entity $e \epsilon E$, $\alpha(e)$ is a finite union of attribute primitives and a subset of $\Omega$. i.e.,

$$\alpha(e) = \langle w_1, w_2, \ldots, w_p \rangle$$

The set of attribute primitives delimited by $\alpha(e)$ is called descriptor, denoted r(e), illustrating the feature profile of knowledge entity, e. Since a knowledge entity (e.g., object) might contain several feature aspects (e.g., characteristic appearances) with each one associated with a descriptor, it might be necessary to characterize a knowledge entity by a disjunction of descriptors, i.e.,

$$\begin{aligned} \text{Feature Profile of } e_i &= \alpha_1(e_i) \cup \alpha_2(e_i) \cup \ldots \cup \alpha_q(e_i) \\ &= r_1(e_i) \cup r_2(e_i) \cup \ldots \cup r_q(e_i) \end{aligned}$$

Each descriptor represents a variation of knowledge entity, $e_i$. In the above expression, entity $e_i$ has q variations. Each

variation is considered as an entity instance of the entity category $e_i$. This concept is useful in calculating conditional projection in section A.3.

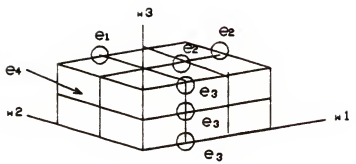
## A.2 N-Cube Model

Definition A.1 An n-cube is an n-dimensional cube with T vertices, where $T=t_1 \cdot t_2 \cdots t_i \cdots t_n$, and $t_i$ is the cardinality of the $i^{th}$ coordinate of the n-dimensional cube. The vertices of the n-cube are labeled with n-tuples of discrete, finite values.

Suppose each dimension (coordinate axis) of the n-cube symbolically represents an attribute variable with coordinate symbolically representing the associated attribute values, then the n-cube can model knowledge domain with vertices denoting the knowledge entities of interest. Since the cardinality of the descriptors associated with entities is not necessarily the same, as depicted in Figure A.1, a knowledge entity might correspond to a vertex such as $e_1$, or multiple vertices such as $e_2$, an edge, such as $e_3$, or a face, such as $e_4$.

## A.3 Converting N-Cube Model To Knowledge Sketch

Since the knowledge hierarchy is connected by node formulas, the issue of knowledge sketch generation becomes how to arrange node formulas. Following the feature of content-based classification in the knowledge hierarchy, we may




$e_1$ = (1, 2, 2)

$e_2$ = (2, 1, 2) ∪ (3, 1, 2)

$e_3$ = (1, 0, -)

$e_4$ = (0, -, -)

Figure A.1 Knowledge Entities in N-Cube Model.

iteratively partition the n-cube into several sub-cubes until each sub-cube represents only a unique primitive entity, and the partition sequences, in return, are used as the needed node formulas. The whole idea is expressed based on the following conventions:

Definition A.2 A partition $\mu$ on a set of vertex T based on certain means, $\beta$, is a collection of disjoint subsets of T with the following properties:

1) each disjoint subset is called a block, denoted $B_i$,
2) $B_i \cap B_j = \phi$, if $i \neq j$,
3) $\cup(B_i) = T$,
4) condition $\beta$ is called partition rule.

Definition A.3 A partition $\mu$ on a n-cube consisting of T vertices is called an n-cube partition, and denoted $\mu(i)$ if the partition rule is conditioned on the $i^{th}$ coordinate axis of the n-cube.

Figure A.2 and Figure A.3 show some realistic aspects of partition and n-cube partition, respectively. It is evident from Figure A.3 that partitioning an n-cube along a specific axis has the same meaning of instantiating the attribute variable associated with that specific coordinate axis. That means if the n-cube partition rule is used as a node formula, the whole partition sequence can be utilized to construct the

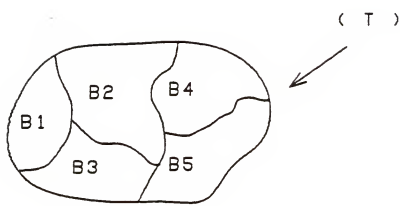



Figure A.2 Partition T into 5 Blocks.

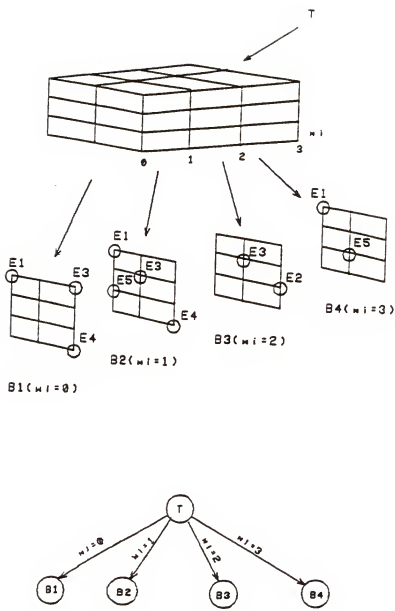



Figure A.3 N-cube Partition Conditioned on $w_1$.

knowledge sketch. The whole idea is summarized as follows.

Conversion process. The knowledge sketch $\pi$ of domain knowledge is obtained on the basis of a sequence of n-cube partition on the n-cube model as follows:

The knowledge hierarchy $\pi$ recursively proliferates itself by associating the terminal node, t, a set of child nodes, $t_1, t_2, .., t_s$, associated with a subset of knowledge entities modeled by blocks, $B_{t1}$, $B_{t2}, .., B_{ts}$, respectively, where the set of blocks are obtained by conducting the partition on the n-cube associated with terminal node t, and the partition rule is conditioned on the $j^{th}$ coordinate of the n-cube where $j^{th}$ coordinate has a cardinality equal to s.

Figure A.4 symbolically shows the knowledge hierarchy generated by this conversion process. Where the first partition rule is the $i^{th}$ coordinate, the second one the $j^{th}$ coordinate, the third one the $k^{th}$ coordinate, and so on. Obviously the sequence of partition rules dictates the path, $(p_1, p_2, .., p_f)$, from the root node to the leaf node, and therefore, determines the shape of the final knowledge structure.

Regarding the diverse properties among a vast amount of application domains, the choice of partition rule tends to be confusing when the size of the knowledge domain is large. In the early research about classification, some work was done

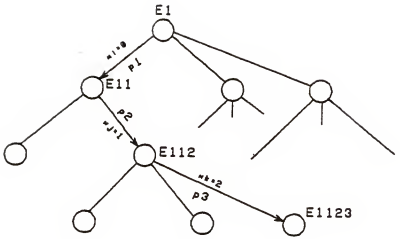



Figure A.4 Partition Rule ($w_i$, $w_j$, $w_k$) Determines the Path ($p_1$, $p_2$, $p_3$) from $E_1$ to $E_{1123}$.

in the area of data analysis to formulate similar type of classification problems into mathematical programming problems with the objective of minimizing cost, processing time, memory space, etc. (Cunningham and Ogilvie, 1972) Here we introduce an algorithm which objectively selects the partition rules based on the concept of conditional projection in conjunction with minimum entropy. We notice that humans are inclined to classify objects following the most noticeable features (Tou and Gonzalez, 1974, Watanabe, 1985). This cognitive process referring to the partitioned n-cube is to select the partition rule along the coordinate axis that represents the most informative attribute. Therefore, after the n-cube partition, the coherence of the sub-domains is increased. We formulate this problem as follows.

Problem formulation. Suppose a semi-organized knowledge domain, $\Phi=(E,\Omega,\Gamma,\mathring{A})$, is modeled by an n-cube where each coordinate axis of the n-cube symbolically represents an attribute variable and the associated coordinates symbolically represent the associated attribute values. The problem of knowledge sketch generation is how, based on concept coherence, properly conduct n-cube partition to transform this n-cube model into the structure of a content associative tree.

To measure the coherence among a group of knowledge entities covered by an n-cube we intuitively define an impurity function as follows (Breiman et al., 1984):

Definition A.4 An impurity function, I, is a function defined on the set of all n-tuples, $(p_1,..,p_j,..,p_n)$, with $p_j \geq 0$, $j=1,..,n$, $\sum_{j=1}^{n} p_j = 1$, satisfying the properties:

(1) I is maximum at the point $[\frac{1}{n}, \frac{1}{n}, ...., \frac{1}{n}]$,

(2) I is minimum at the points (1,0,..,0), (0,1,0,..,0), ..., (0,...,0,1).

(3) I is a symmetric function of $p_1,..,p_j,..,p_n$.

The heuristic in this impurity function is obvious. If we interpret each tuple element $p_j$ as the ratio of knowledge entity (including its variations or entitiy instance) of $j^{th}$ entity category to the total number of knowledge entities within the entity space, then the n-tuple with property (1) denotes the situation that there are n entities coming from n different categories, hence, the impurity measure reaches maximum. Property (2) means all the n entity instances are from the same category, thus the impurity measure is minimum.

Entropy function and several other functions fulfill the conditions specified in the intuitive impurity function (Schank, 1975, Breiman et al., 1984). As a matter of fact, the entropy function has well been recognized as a good parameter measuring the randomness of a universe (Watanabe, 1985), so it is natural to adopt this function as our impurity function, and thus we have the following definitions:

Definition A.5 For an n-cube containing a cluster of entity instances with the entity category distribution being expressed in the form of an n-tuple, the impurity measure of this n-cube is defined as the entropy value associated with that n-tuple.

Definition A.6 The impurity reduction due to a n-cube partition $\mu(j)$ is defined as:

$$\Delta I_j = I(B) - \sum_{i=1}^{k} I(B_i) \cdot p_i \qquad (A.1)$$

where $I(B)$ is the impurity measure of the n-cube before partition, $I(B_i)$ is the impurity measure of the sub-block $B_i$ after partition, $p_i=1/k$, with k is the cardinality of the $j^{th}$ coordinate axis. (Here $p_i$ has meaning of probability with an even probability density function.)

Based on the principle of minimum entropy (Watanabe, 1985), the most informative feature will be the one which can create the maximum impurity reduction. Obviously, following equation (A.1), the key element in making a selection is how to measure the impurity value associated with a specific block $B_i$. This measure can be obtained based on the concept of projected distribution interpreted as below.

Definition A.7 In an n-cube, the projected distribution of entity $e_j$ onto attribute variable $w_i$ is defined as:

$$P_{wi}^{ej} = \left(\frac{a_1^{ej}}{b_1}, \ldots, \frac{a_i^{ej}}{b_i}, \ldots, \frac{a_k^{ej}}{b_k}\right)$$

$$= \left( p_{wi1}^{ej}, \ldots, p_{wij}^{ej}, \ \ldots, \ p_{wik}^{ej} \right) \qquad (A.2)$$

where $a_i^{ej}$ and $b_i^{ej}$ are the number of entity instances of $e_j$ and the total number of entities in the n-cube projected to the $i^{th}$ attribute value of attribute variable $w_i$, respectively, k is the cardinality of attribute $w_i$, $p_{wij}^{ej}$ is defined as the conditional projection of $e_j$ conditioned on $i^{th}$ attribute with $j^{th}$ attribute value. This projected distribution has the meaning of possible coverage on a specific entity $e_j$ characterized by attribute variable $w_i$.

In case the significance of each attribute has been fine-tuned by the associated attribute weighting factor as specified in section 2.3.1, this modulation effect can be included by making $a_i^{ej}$ as the summation of attribute weighting factor of entity $e_j$ in the n-cube projected to the $i^{th}$ attribute value of attribute variable $w_i$, and $b_k$ is the summation of all attribute weighting factors associating the entity covered in the n-cube projected to the $i^{th}$ attribute

value of attribute $w_i$.

By the same token, if there are m types of entities in the n-cube, we can calculate their projected distribution respectively as:

$$
\begin{aligned}
p^{e1}_{wi} &= ( p^{e1}_{wiq} \mid q=1,..,k) = (p^{e1}_{wi1}, p^{e1}_{wi2}, \ldots, p^{e1}_{wik}) \\
p^{e2}_{wi} &= ( p^{e2}_{wiq} \mid q=1,..,k) = (p^{e2}_{wi1}, p^{e2}_{wi2}, \ldots, p^{e2}_{wik}) \\
&\ \ \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
p^{em}_{wi} &= ( p^{em}_{wiq} \mid q=1,..,k) = (p^{em}_{wi1}, p^{em}_{wi2}, \ldots, p^{em}_{wik})
\end{aligned}
\qquad \text{(A.3)}
$$

From equation (A.3) we can form the tuple

$$
P_{wij} = (p^{e1}_{wij}, p^{e2}_{wij}, \ldots\ldots, p^{em}_{wij}) \qquad \text{(A.4)}
$$

with its elements taken from the projection of each entity $e_j$ conditioned on $i^{th}$ attribute with $j^{th}$ attribute value. The entropy of this conditioned distribution $P_{wij}$, denoted $H(e|w_{ij})$, can be calculated as:

$$
H(e|w_{ij}) = \sum_{r=1}^{m} [-p^{er}_{wij} \cdot \log(p^{er}_{wij})] \qquad \text{(A.5)}
$$

This value, via definition A.5, is the impurity measure of the n-cube partitioned block conditioned on $i^{th}$ attribute with $j^{th}$ attribute value. Under the assumption that each attribute

value of $w_i$ has equal opportunity of occurrence, the average impurity measure of the n-cube partition conditioned on (partition rule) $w_i$ is:

$$\bar{H}(e|w_i) = \frac{1}{k} \left( \sum_{j=1}^{k} H(e|w_{ij}) \right) \qquad (A.6)$$

Following equation (A.1), the impurity reduction conditioned on $w_i$ can be calculated by

$$\Delta I|_{wi} = I - \bar{H}(e|w_i) \qquad (A.7)$$

Similarly, we can calculate the impurity reduction conditioned on other attribute variables. Since I is constant for each $w_i$, the most informative attribute is the one which can deliver the smallest average entropy of the projected distribution obtained from equation (A.6), and the selected $w_i$ will be the partition rule for conducting n-cube partition. This partition procedure can be summarized as follows:

Objective partition procedure: PART(N,E,Ω)

1. If block N contains only single primitive entity, then return, otherwise, let $e_1,\ldots,e_j,..,e_m \epsilon E$ be included entities instances, and $w_1,..,w_i,..,w_n \epsilon \Omega$ be the associated attributes, following equation (A.2), calculate the projection distribution of entity $e_j$ on attribute

variable $w_i$.

2. If $j < m$, repeat step 1, else continue.
3. Based on equation (A.3), (A.4), (A.5), (A.6) calculate the average entropy of the projected distribution of domain universe conditioned on attribute $w_i$, denoted $H(E|w_i)$.
4. If $i < n$, go to step 1, else continue.
5. Set partition rule $\beta=w_r$, where $H(E|w_r) < H(E|w_i)$, $\forall i$, $i \neq r$.
6. Conduct n-cube partition based on the partition rule $\beta$, and proliferate the knowledge hierarchy following the conversion process given in section 2.4.3, then go to step 1.

Quinlan (Quinlan, 1983) has a similar approach but based on a different model. If the objective of knowledge utilization is classification-based, then this objective partition procedure has its own value, especially when inference efficiency is taken into account. In case that the main concern of knowledge hierarchy is for counselling usages, since domain semantics as well as human's expertise are not involved in the process of partition rule selection, the intermediate entities so obtained generally do not retain comprehensible meaning, thus, its applicability will be limited. For this kind of situation, other subjective means are preferred.

## REFERENCES


Barnett, J. A., "Computational Methods for A Mathematical Theory of Evidence," Proc. of 7th Int. J. Conf. on Artificial Intelligence, pp. 868-875, 1981.

Barr, A., Feigenbaum, E. A., (Eds.), The Handbook of Artificial Intelligence, William Kaufmann, Inc., Los Altos, CA, 1981.

Bayegan, H. M., and Asa, E., "An Integrated System for Interactive Editing of Schematics, Logic Simulation and PCB Layout Design," Proc. 14th Design Automation Conference, pp. 1-6, 1977.

Begg, V., Developing Expert CAD Systems, Unipub, New York, NY, 1984.

Brachman, R. J., Fikes, R. E., Levesque, H. J., "KRYPTON: A Functional Approach to Knowledge Representation," IEEE Computer 16, pp. 67-73, 1983.

Brachman, R. J., Levesque, H. J., (Eds), Readings in Knowledge Representation, Morgan Kaufmann Pub. Inc., Los Altos, CA, 1985.

Brachman, R. J., Schmolze, J. G., "An Overview of the KL-ONE Knowledge Representation System," Cognitive Science, 9, pp. 171-216, 1985.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., Classification and Regression Trees, Wadsworth International Group, CA, 1984.

Bryant, R. E., "A Switch-Level Model and Simulator for MOS Digital Systems," IEEE Trans. on Computers, Vol. C-33, NO. 2, pp. 160-177, 1984.

Chandrasekaran, B., Sanjay, M., "Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related System," Advanced in Computers, Vol. 22, pp. 217-293, 1983.

Chandrasekaran, B., "From Numbers to Symbols to Knowledge Structures: Pattern Recognition and Artificial Intelligence Perspectives on Classification Task," in Pattern Recognition in Practice II, E. Gelsema, L. N. Kanal (Eds.), pp. 547-559, 1984.

Chandrasekaran, B., Tanner, M. C., "Uncertainty Handling in Expert Systems: Uniform vs. Task-specific Formalisms," in Uncertainty in Artificial Intelligence, L. N. Kanal, J. F. Lemmer (eds), North-Holland, Amsterdam, pp. 35-46, 1986.

Chang, L. C., Tou, J. T., "MEDIKS- A Medical Knowledge System," IEEE Trans. on System, Man, and Cybernetic, Vol. SMC-14, NO. 5 Sept/Oct, pp.746-750, 1984.

Charniak, E., McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley, Reading, MA, 1985.

Cheeseman, P. C., "A Method of Computing Generalized Bayesian Probability Values for Expert Systems," Proc. of 8th Int. J. Conf. on Artificial Intelligence, pp. 198-202, 1983.

Cheeseman, P. C., "In Defence of Probability," Proc. of 9th Int. J. Conf. on Artificial Intelligence, Los Angeles, pp 1002-1009, 1985.

Cheng, J. M., Knowledge Base For Consultation And Image Interpretation, Ph.D. dissertation, University of Florida, 1983.

Cohen, P. R., Grinberg, M. R., "A Theory of Heuristic Reasoning about Uncertainty," The AI Magazine, Summer, pp. 17-24, 1983.

Cunningham, K. M., Ogilvie, J. C., "Evaluation of Hierarchical Grouping Techniques: A Preliminary Study," Computer Journal (15), pp. 209-213, 1972.

Dai, M. T., Tou, J. T., "KETOG- A Knowledge Engineering Tool for Organizing and Constructing the Knowledge Base in Pattern-Directed Expert Systems," Proc. of the Int. Computer Symposium, pp. 647-652, 1988.

Daniel, M., Gwyn, C., "CAD Systems for IC Design," IEEE Transactions on CAD of IC and System, Vol. CAD-1, No. 1, January, 1982.

Dejong, K., "Expert Systems in the ATE Arena," AUTOTESTCON '85 Symposium Proceedings, pp. 129-132, 1985.

Deo, N., Graph Theory with Application to Engineering and Computer Science, Prentice-Hall, Edgewood Cliffs, NJ, 1974.

Desoer, C. A., Kuh, E. S., Basic Circuit Theory, McGraw-Hill Book Co., New York, NY, 1969.

Doyle, J., "A Truth Maintenance System," Artificial Intelligence, 12, pp. 231-272, 1979.

Driankov, D., "A Many-valued Logic for Belief/Disbelief Pairs," in Methodologies for Intelligent Systems, Z. W. Ras, M. Zemankova (Eds.), pp. 25-32, Elsevier Science Publishing Co., Inc. New York, NY, 1987.

Duda, R. O., and Hart, P. E., Pattern Classification and Scene Analysis, John Wiley & Sons, Inc., New York, NY, 1973.

Duda, R., Hart, P., Nilsson, N., "Subjective Bayesian Methods for Rule-based Inference Systems," in Reading in Artificial Intelligence, B. L. Webber, N. J. Nilsson (Eds.), pp. 192-199, Tioga Publishing Co., Palo Alto, CA, 1981.

Findler, N. V., (Ed.), Associative Networks, Representation and Use of Knowledge by Computers, Academic Press, New York, NY, 1979.

Garvey, T. D., Lowrance, J. D., Fischler, M. A., "An Inference Technique for Integrating Knowledge From Disparate Sources," Proc. of 7th Int. J. Conf. on Artificial Intelligence, pp. 319-325, 1981.

Gero, J., (Ed.), Expert Systems in Computer-aided Design, Elsevier Science Publishers, B.V., New York, NY, 1987.

Gevarter, W. B., Intelligent Machines, An introductory Prospective of Artificial Intelligence and Robotics, Prentice-Hall, Edgewood Cliffs, NJ, 1985.

Gordon, A. D., Classification, Chapman and Hall, New York, NY, 1981.

Groen, F. C. A., and Munster, R. J. V., "Computer Aided Analysis of Schematic Diagrams," in Pattern Recognition in Practice II, E. S. Gelsema, L. N. Kanel (Eds.), pp. 363-372, Elsevier Science Publisher, B.V., New York, NY, 1986.

Hayes-Roth, F., "The Role of Partial and Best Matches in Knowledge Systems," in Pattern-Directed Inference Systems, pp. 557-576, Academic Press, New York, NY, 1978.

Hayes-Roth, F., Waterman, D. A., Lenat, D. B., (Eds.), Building Expert Systems, Addison-Wesley, Reading, MA, 1983.

Hewitt, C., "PLANNER: A Language for Proving Theorems in Robots," Proc. of 2nd Int. Conf. on Artificial Intelligence, pp. 105-112, 1971.

Horstman, P. W., "Expert Systems and Logic Programming for CAD," VLSI Design, November, pp. 37-46, 1983.

Jumarie, G. M., Subjectivity, Information, Systems, An Introduction to A Theory of Relative Cybernetics, Gordon and Breach Science Publishers, New York, NY, 1986.

Kanal, L. N., Lemmer, J. F., (Eds), Uncertainty in Artificial Intelligence, North-Holland, Amsterdam, 1986.

Kerschberg, I., (Ed.), Expert Database System, The Benjamin/Cummings Publishing Co., Inc., Menlo Park, CA, 1986.

Kim, J., Pearl, J., "A Computational Model for Combined Causal and Diagnostic Reasoning in Inference Systems," Proc. of 8th Int. Joint. Conf. on Artificial Intelligence, pp. 190-193, 1983.

Knowles, R., Automatic Testing Systems and Applications, McGraw-Hill Book Co., New York, NY, 1976.

Kosko, B., "Fuzzy Knowledge Combination," Int. J. of Intelligent System, Vol. I, pp. 293-320, 1986.

Kowalik, J. S., Coupling Symbolic and Numerical Computing in Expert Systems, Elsevier Science Publishers, B.V., New York, NY, 1986a.

Kowalik, J. S., (Ed.), Knowledge Based Problem Solving, Prentice-Hall, Edgewood Cliffs, NJ, 1986b.

Minsky, M. "A Framework for Representing Knowledge," in The Psychology of Computer Vision, Winston, P., (Ed.), McGraw-Hill Book Co., New York, NY, 1975.

Negoita, C. V., Ralescu, D., Simulation, Knowledge-based Computing, and Fuzzy Statistics, Van Nostrand Reinhold, Co., New York, NY, 1987.

Nilsson, N. J., Principles of Artificial Intelligence, Tiogo Pub. Co., Palo Alto, CA, 1980.

Odawara, G., Kurishima, S., Aoyama, H., and Kanaya, Y., "PAS-CIP: An Interactive Logic Design System," Proc. 18th Design Automation Conference, pp. 443-449, 1981.

Pawlak, Z., "Rough Sets," Int. J. of Comp. and Information Sciences, Vol. 11, No. 5, Oct. pp. 341-356, 1982.

Pearl, J., "A Constraint-propagation Approach to Probabilistic Reasoning," Proc. of the Workshop on Uncertainty and Probability in Artificial Intelligence, pp. 185-192, 1985.

Peebles, P. Z. Jr., Probability, Random Variables, and Random Signal Principles, McGraw-Hill Book Co., New York, NY, 1980.

Porter, K. A. Jr., "AI Applications to Automatic Testing: Trend for the Future," AUTOTESTCON '87 Symposium Proceedings, pp. 377-382, 1987.

Quinla, R. J., "Learning Efficient Classification Procedure and Their Application to Chess End Games," in Machine Learning, An Artificial Intelligence Approach, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), pp. 463-482, Tioga Pub. Co., Palo Alto, CA, 1983.

Reiter, R., "A Theory of Diagnosis From First Principles," Artificial Intelligence, Vol. 32, pp. 57-95, 1987.

Rich, E., Artificial Intelligence, McGraw-Hill Book Co., New York, NY, 1983.

Robinson, J. A., "A Machine-oriented Logic Based On The Resolution Principle," J. of ACM, Vol. 12, pp. 23-41, 1965.

Rollinger, C. R., "How To Represent Evidence-Aspects of Uncertainty Reasoning," Proc. of 8th Int. Joint Conf. on Artificial Intelligence, pp.358-361, 1983.

Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checker II-- Recent Progress," IBM J. of Research and Development 11, 6, pp. 601-617, November 1967.

Schank, R. C., Conceptual Information Processing, North-Holland, Amsterdam, 1975.

Schank, R. C., Abelson, R. P., Scripts, Plans, Goals, and Understanding, Lawrence Erlbaum, Hillsdale, NJ, 1977.

Shafer, G., "Belief Functions and Parametric Models," J. of Royal Statistical Society, B, 44, pp. 332-352, 1982.

Shafer, G., "The Combination of Evidence," Int. J. of Intelligent Systems, Vol. 1, pp. 155-179, John Wiley & Sons, Inc., New York, NY, 1986.

Shapiro, S., Smith, R. J., Handbook of Design Automation, Prentice-Hall Edgewood Cliffs, NJ, 1984.

Shortliffe, E. H., Computer-based Medical Consultation-MYCIN, American Elsevier Publishing Co., Inc., New York, NY, 1976.

Sussman, G. J., Stallman, R. M., "Heuristic Techniques in Computer-Aided Circuit Analysis," IEEE Trans. on Circuit and System, vol. CAS-22, no. 11, Nov. 1975.

Szolovits, P., "Types of Knowledge as Base for Reasoning in Medical AI Programs," in Artificial Intelligence in Medicine, Lotto, I., and Stehanelli, M., (Eds.), pp. 209-226, North-Holland, Amsterdam, 1985.

Szolovits, P., Pauker, S., "Categorical and Probabilistic Reasoning in Medical Diagnosis," Artificial Intelligence, Vol. 11, pp. 115-144, 1978.

Tou, J. T., "MEDIKS- A Medical Knowledge System," Proc. of the 31th Annual Conference on Engineering in Medicine and Biology, Atlanta, GA, 1978a.

Tou, J. T., "Design of A Medical Knowledge System for Diagnostic Consultation and Clinical Decision-Making," Proc. of the Int. Computer Symposium, Vol. 1, pp. 80-99, 1978b.

Tou, J. T., "Application of Pattern Recognition to Knowledge System Design and Diagnostic Inference," in Pattern Recognition Theory and Applications, J. Kittler, K. S. Fu, L. F. Pau (Eds.), pp. 413-429, D. Reidel Publishing Co. New York, NY, 1982.

Tou, J. T., "Knowledge Engineering Revisited," Int. J. of Computer and Information Science, Vol. 14, No. 3, pp. 123-133, 1985.

Tou, J. T., Cheng, J. M., "AUTORED: An Automated Electronic Diagram Reading Machine," Proc. of 1983 IEEE Int. Conf. Computer-Aided Design, 1983a.

Tou, J. T., Cheng, J. M., " Design of A Knowledge-Based Expert System for Application in Agriculture," IEEE Trends and Applications in 1983, pp. 19-21, 1983b.

Tou, J. T., DePree, R. W., "Telebrowing System and Its Applications", Proc. 1978 European Computing Congress, London, England, 1978c.

Tou, J. T., Depree, R. W., "Medical Consultation via Telebrowsing," Proc. of Int. Conf. on Medical Comp. Berlin, 1979.

Tou, J. T., Gonzalez, R. C., Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.

Tou, J. T., Huang, C. L., Li, W. H., "Design of a Knowledge-based System for Understanding Electronic Circuit Diagrams," Proc. of First Conf. on Artificial Intelligence Applications, pp. 302-308, 1984.

Tou, J. T., Huang, C. L., "Knowledge-based Functional-Symbol Understanding in Electronic Circuit Diagram Interpretation," pp. 652-657, 1984.

Tou, J. T., Li, W. H., Fan, K. C., Huang, C. L., "Knowledge-based Approach for the verification of CAD Database Generated by an Automatic Schematic Capture System," Proc. of the 24th Design Automation Conference, June, pp. 302-308, 1987.

Tou, J. T., Sutton, F. R., "SEFIRE: A Sequential Feedback Interactive Retrieval System," in Information Systems, COINS IV, J. T. Tou (Ed.), pp.197-217, Plenum Press, New York, NY, 1974.

Watanabe, S., Pattern Recognition: Human and Mechanical, John Wiley & Sons, Inc., New York, NY, 1985.

Waterman, D. A., A Guide To Expert Systems, Addison-Wesley, Reading, MA, 1986.

Waterman, D. A., Hayes-Roth, F., (Eds.), Pattern-Directed Inference Systems, Academic Press, New York, NY, 1978.

Winston, P. H., Artificial Intelligence, 2nd ed. Addison-Wesley, Reading, MA, 1984.

Wirth, N., Algorithm + Data Structure = Programs, Prentice-Hall, Edgewood Cliffs, NJ, 1976.

Wojciki, A. S., "Formal Design Verification of Digital Systems," Proc. of the 20th Design Automation Conference, June, pp. 312-318, 1983.

Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," IEEE Trans. System and Cybernetics, SMC-3;1 pp. 28-44, 1973.

Zadeh, L. A., "Fuzzy Sets as A Basis for A Theory of Possibility," Fuzzy Sets and Systems, pp. 3-28, 1978.

## BIOGRAPHICAL SKETCH

Ming-Tsair Dai was born in Kaohsiung, Taiwan, Republic of China (ROC), on March 2, 1954. He received his B.S. degree from the National Chiao Tung University, Taiwan, ROC, in 1977, and his M.S. degree from the University of Colorado, Boulder, Colorado, in 1981, both in electrical engineering. From 1977 to 1979, he was working at the Chung Shan Institute of Science and Technology, Taiwan, ROC, as a circuit engineer. In 1982 he was promoted to system engineer. Since December 1985, he has been working toward the Ph.D. degree in electrical engineering at the University of Florida, Gainesville, FL.

His research interests include pattern recognition, computer vision, artificial intelligence, and expert systems.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Julius T. Tou, Chairman
Graduate Research Professor of
Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Peyton Z. Peebles, Jr.,
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John Staudhammer
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Leon W. Couch II
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yuan-Chieh Chow
Professor of Computer and
Information Sciences

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December 1989

Winfred M. Philips
Dean, College of Engineering

Madelyn M. Lockart
Dean, Graduate School